Energy-aware Software

POLITECNICO DI TORINO

SCUOLA DI DOTTORATO

Dottorato in Ingegneria Informatica e dei Sistemi – XXVI ciclo

Tesi di Dottorato

Energy-aware software

Luca Ardito

Tutore
Prof. Maurizio Morisio

Coordinatore del corso di dottorato
Prof. Pietro Laface

Dicembre 2013

To my Grandmother
"Mina"

Summary

Luca Ardito has focused his PhD on studying how to identify and to reduce the energy consumption caused by software. The project concentrates on the application level, with an experimental approach to discover and modify characteristics that waste energy. We can define five research goals:

- **RG1. Is it possible to measure the energy consumption of an application?** Measuring the energy consumption of an electronic device (PC, mobile phone, etc.) is straightforward, but several applications coexist on it, possibly with very different energy needs. Usage profiles for applications are certainly important too. We will consider the most common platforms (Windows, Linux, Mac Osx)[1].

- **RG2. Could Energy Efficiency be considered as a software non-functional requirement?** Research has increasingly focused on improving the Energy Efficiency of hardware, but the literature still lacks in quantifying accurately the energy impact of software. This research goal is strictly related to the following one.

- **RG3. Is it possible to profile the energy consumption of a software application?** An empirical experiment could assess quantitatively the energetic impact of software usage by building up common application usage scenarios and executing them independently to collect power consumption data.

- **RG4. Is there a relationship between the way a program is written and its energy consumption?** The same application, at the code level, can be written in different ways. Here the question is if the different ways have impact on energy consumption. The code should be considered at two levels: source code (programmer) and object code/byte code (compiler). [88]

---

[1] `https://01.org/powertop/` Last visited: 2014/01/02

- **RG5. Is it possible to use the energy consumption information to trigger self-adaptation?** A software application could automatically modify its behaviour in order to reduce its energy consumption.

Acknowledgements

Reaching this point, I can hardly believe I have achieved my goal. During my master's thesis, written between the Department of Computer and Control Engineering of Politecnico di Torino and Telecom Italia Labs, I discovered the world of research. From that moment I decided to apply for the PhD programme at Politecnico di Torino. When I realised that I had been selected for the PhD, it did not seem real, and I saw the time of delivery of the PhD dissertation as far away and not very certain. During these years I learned a lot, and had the opportunity to participate in many conferences around the world (I still clearly remember how I was scared during my first "talk" in Athens!). I came to know the world of research: a very new environment for me. For this reason I would like to thank my supervisor: Professor Maurizio Morisio, who motivated me during these three years, and especially as he gave me the opportunity to deal with many situations that at the time seemed to me to be far above my abilities. Thanks also to Professor Marco Torchiano, who is always ready to help me when I have a problem. And how can we forget all my PhD mates: Federico, Antonio, Giuseppe Rizzo, Matteo, Andrea, Edoardo, Marco, Oscar, Cristhian, Ali, Najeeb, and Giuseppe Procaccianti. We always managed to group ourselves against Procaccianti on any occasion (especially Federico!), in that way making the atmosphere more pleasant and less stressful (especially when we tried to give Procaccianti all our work to do... but probably this is not the right time to reveal these secrets!). A special thanks goes to the team of Telecom Italia Labs composed of Carlo Licciardi, Marco and Luca (great travel companions during the Californian trips) and Lucia. I have never stopped cooperating with them since I started my master's thesis in 2009: you are among the first to have believed in me, thank you. A special thanks goes to Ingrid, who put up with me during all these years. You had (and you have) great patience to be around me when things were not going exactly as I expected (therefore almost always!). Thank you *Tatina*. I cannot forget to thank the people who lived at No.14: Stefano, Fonta, Filippo, Paolo, and Emiliano. At 14 Via Gioberti, there was always something fun to do! Thanks to my friends Stefano, Alan, Marco, Andrea S., Veronica, Andrea G., Claudia, Diego B., Silvia, Maria, Lucia, Loredana, Diego R., who have encouraged, entertained, supported me through the dark times, celebrated with me through the good, who have been brilliant and understanding when I needed them to be, I take this opportunity to thank you. Last but not least I thank my family, Mum Pinuccia, Dad Marco, and Grandparents Beppe, Battista and Ettorina. All you have ever done is the impossible for me and you gave me the opportunity to tackle this path. Without your support (both moral and material), it would never have been possible. I will never cease to give thanks for you.

List of Tables

List of Figures

Contents

Chapter 1

Introduction

Greenhouse gas emissions and Power Consumption are problems of relevant importance. The EU has identified the ICT industry as the most important actor in the 20-20-20 challenge: it is estimated that the ICT sector can reduce global emissions from all sectors of 15% by 2020. However, the environmental impact of ICT is not only positive: ICT was responsible for 10% of energy demand in 2010 and for 2% of global CO2 emissions (expected to grow to 8% in 2020) [103]. The energy impact of ICT was often analysed in the context of embedded systems, where energy efficiency is a key point; so far, little research on collection and analysis of power consumption data, at the application level, exists [22] [63]. As said above, energy consumption is usually analysed at the hardware and low-level software level. Little research is available on the software application level. The project concentrates on the application level, with an experimental approach to discover and modify characteristics that waste energy.

1.1 Research Goals

This dissertation is written according to the following research goals. It also provides a detailed analysis of the state of the art, which defines a taxonomy to classify papers; all this with the aim of being able to repeat the same work after a few years to compare how the data vary.

RG1. Is it possible to measure the energy consumption of an application?

Measuring the energy consumption of an electronic device (PC, mobile phone, etc.) is straightforward, but several applications coexist on it, possibly with very different energy needs. Usage profiles for applications are certainly important too. The most common platforms are Windows, Linux, and Mac Osx [1]. Within an application, we can identify two layers: the application layer and the platform layer, in turn divided into the Operating system and devices. The first question is if it is meaningful to trace the consumption of applications to layers. If this is not feasible, then energy consumption (and the efforts to reduce it) should consider the platform layer only [23].

RG2. Could Energy Efficiency be considered as a software non-functional requirement?

Research has increasingly focused on improving the Energy Efficiency of hardware, but the literature still lacks in quantifying accurately the energy impact of software. This research goal is strictly related to the following one.

RG3 Is it possible to profile the energy consumption of a software application?

An empirical experiment could assess quantitatively the energetic impact of software usage by building up common application usage scenarios (e.g.: Skype call, Web Navigation, Word writing) and executing them independently to collect power consumption data. Based on the use of PCs, the resources are used differently. It is necessary to identify the profile of use of the systems and verify if and how energy consumption varies.

RG4. Is there a relationship between the way a program is written and its energy consumption?

The same application, at the code level, can be written in different ways. Here the question is if the different ways have impact on energy consumption. The code should be considered at two levels: source code (programmer) and object code/byte code (compiler) [88].

- **RG4.1. Can we identify energy smells in the source code?** A code smell is a characteristic of a source code that is considered negative and that could represent a cue of a fault. Smells can be both at design level and at code level. With energy smell we want to define a particular code or design pattern, which can impact on the energy efficiency of an application.
- **RG4.2. Can we refactor energy smells in the source code?** Often a smell is linked to a refactoring action, that is proposed to fix the smell and

[1] `https://01.org/powertop/` Last visited: 2014/01/02

improve properties of the software module, such as maintainability, comprehensibility, reliability, performance. If it is possible to identify, automatically or manually, the energy smells, a refactoring action should be defined for each smell found.

- **RG4.3. Is it possible to analyse statically the source code of an application in order to classify its energy efficiency?** The static code analysis should identify energy inefficient situations of the source code. If so, we can evaluate the energy efficiency of the source code that is analysed.

RG5. Is it possible to use the energy consumption information to trigger self-adaptation?

A software application could automatically modify its behaviour in order to reduce its energy consumption.

1.2 Methodology

The methodology utilised is Empirical Software Engineering. The Empirical Software Engineering (EMSE) research can be applied using two paradigms: qualitative and quantitative research. Qualitative research studies objects in their natural settings, and phenomena are explained by eliciting explanations from people. This approach is for construction of perspective-based results and reflects the question "Why?". Quantitative research focuses on the quantification of a relationship or a comparison of two or more groups. This type of research answers the question "How?". The two approaches are complementary and are often mixed or alternated in research. Both approaches provide a set of methodological tools defined by Shull et al. [91] as a set of organising principles around which empirical data is collected and analysed. These tools are called empirical strategies by Wohlin et al. [106], kinds of empirical studies by Juristo et al. [52] and "research methods" by Shull et al. [91]. The first two authors identify three methodological tools (experiments, surveys, case studies), and the latter one adds two more tools (ethnographies and action research). Below are presented the tools as defined by Shull et al. [91].

- Survey: it is used to identify the characteristics of a broad population of individuals. It is conducted through the use of questionnaires, structured interviews, or data logging techniques. One important step in survey researches is the identification of a representative subset of the population, since usually it is not possible to poll every member.
- Case study: it investigates a phenomenon within its real-life context, offering in-depth understanding of how and why certain phenomena occur, thus

investigating cause / effect relationships (qualitative research). Exploratory case studies are used as initial investigations of some phenomena to derive new hypotheses and build theories, while confirmatory case studies are used to test existing theories.

- Controlled experiment: it is an investigation of a testable hypothesis where one or more independent variables are manipulated to measure their effect on one or more dependent variables. Each combination of values of the independent variables is a treatment. True experiments are not always possible in SE (e.g., full randomisation is often not achievable in real contexts); in that case quasi-experiments can be conducted (for instance, the subjects are not assigned randomly to the treatments).

- Ethnography: this method focuses on the sociological aspects. Ethnographies study a community of people to understand how the members of that community make sense of their social interactions. For software engineering, ethnography can help to understand how technical communities build a culture of practices and communication strategies that enables them to perform technical work collaboratively. A special form of ethnography is participant observation, where the researcher becomes a member of the community being studied for a period of time.

- Action Research: in Action Research, the researchers attempt to solve a real-world problem while simultaneously studying the experience of solving the problem, intervening inside the real contexts to improve the situation.

This work has been carried out by using: Case Study, Controlled Experiment, and Action Research.

1.3 Thesis Outline

The contributions of this PhD program can be summarised in four points:

1. Literature review: described in Chapter 2;

2. Analysis of energy consumption as a non-functional requirement: described in Chapter 3;

3. Characterisation of the energy consumption of software applications and systems with an empirical approach; identification of software characteristics that waste energy: described in Chapter 4;

Finally, Chapter 5 summarises the work done.

Chapter 2

Background

The rapid growth and significant development of IT systems has started to cause an increase of worldwide energy consumption [103]. This issue moved technology producers, information systems managers, and researchers to deal with energy consumption reduction in terms of:

- Global CO2 footprint;
- Consumption of data centres;
- Reduced battery life of portable devices;
- Economic impact of a new business model, which aims at greening everything.

As well described in [59], IT producers are forced to manage the product lifecycle through legislation, but also users are becoming concerned about the environmental implications from the use of IT. This area of research is called GREEN IT, which “refers to environmentally sustainable computing or IT[1]” and is defined as “The study and practice of designing, manufacturing, using, and disposing of computers, servers, and associated subsystems such as monitors, printers, storage devices, and networking and communications systems efficiently and effectively with minimal or no impact on the environment” [74]. Murugesan in [74] also expresses the benefits introduced by Green IT: “Green IT benefits the environment by improving energy efficiency, lowering greenhouse gas emissions, using less harmful material, and encouraging reuse and recycling”. Taking into account the research areas identified in [19] and according to [74], the main topics related to Green IT, are:

- Design for environmental sustainability: “balancing energy and resource savings by ICT and energy and resource consumption of ICT” [76] and “making business operations, buildings, and other systems energy-efficient” [75];

[1] `http://en.wikipedia.org/wiki/Green_computing` Last Visited: 11 Apr 2013

- Energy-Efficient Computing: the efficient use of resources in terms of energy-aware algorithms;
- Power Management: a set of HW/SW techniques that optimise the management of power resources in computer systems, portable devices, and data centres.
- Data centre design: eco-friendly devices that improve energy efficiency and energy conservation of data centres (i.e., energy-efficient mechanical and electrical systems, green power, use of natural light, etc.);
- Virtualization: “the faithful reproduction of an entire architecture in software, which provides the illusion of a real machine to all software running above it” [56];
- Disposal and recycling management: managing e-waste, and limiting planned obsolescence upgrading devices instead of replacing them;
- Regulatory Compliance: Regulatory requirements and legislative actions tend to force acceptance of a technology or practice in situations where this would not occur. The existence of certain rules on sustainability in IT standards can lead to the adoption of some green IT initiatives [71];
- Green metrics, tools and methodology assessments: software tools for collecting or simulating, analysing, modelling, reporting energy consumptions, environmental risk management, environmental impact, and greenhouse gas emissions; platforms for eco-management, emission trading, or ethical investing [74].

Green IT involves many areas and stakeholders, starting from governments, through new business models and R&D, to different technical fields. IT can also be used to monitor energy consumption such as: heating systems in buildings, fuel efficiency in cars or smart grids implementation. So IT is involved in worldwide energy reduction, and in reducing its energy consumption as well. The main contribution of this work is to summarise evidence available in the literature about:

- Measuring techniques for energy consumptions in IT systems;
- IT energy consumption trend according to our taxonomy;
- Methodologies to reduce energy consumption of IT;

The goal is to offer a clearer picture of the state of the art in this field, and to highlight areas where evidence is missing as well [9]. This chapter is organized as follows:

- Section 2.1 proposes our taxonomy.
- Section 2.2 deals with the energy metrics, which are used to characterise more quantitatively what “greenness” means at each layer of the taxonomy.
- Section 2.4 describes some guidelines that can improve energy efficiency in IT systems.
- Section 2.5 gives our conclusions.

2.1 Taxonomy

We define the following taxonomy to organize IT and energy consumption. We consider two orthogonal dimensions, the time axis (or IT energy lifecycle) and the space axis (IT elements and infrastructure).

2.1.1 IT Energy Life Cycle

Inspired by [32] we propose the energy life cycle in Figure 2.1. The activities in the lifecycle are design of the IT product, manufacture, transport (includes packaging and distribution), use, disposal, and possibly recycling. All these activities consume materials and energy, with related emissions.

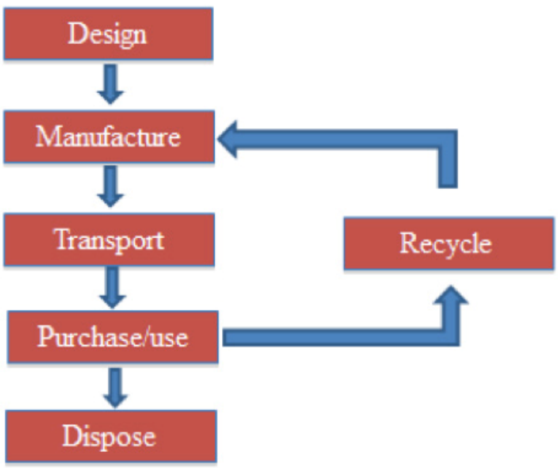

Figure 2.1. The time axis: Energy Life Cycle

2.1.2 IT Elements and Infrastructure

On the spatial axis, we consider elements or nodes (e.g. PC, peripheral devices and mobile phones) and their connections (cables, wireless links etc.) to build infrastructures (e.g. PC networks, the Internet, data centres, the cloud). In a node, we define different layers (starting from hardware to application) as defined later (Figure 2.2)

- Network element: this element considers network equipment (e.g. Network Interface Card, router and gateways) and protocols, which means everything is related to connectivity.

- A node element consists of three layers:

  - Hardware layer: this layer considers CPUs, GPUs, and storage (memory, disks).
  - Operating System layer: this layer considers software programs implementing the traditional operating system services (file and memory management, task scheduling, I/O management). The key issue here is power management.
  - Application layer: this layer considers software to implement user level services. Key issues considered here are the energy efficiency of algorithms and software architectures.

- Infrastructure element: This element considers many nodes connected by a network to define a larger entity capable of offering higher-level computation services. Entities of this kind are data centres, web farms, and cloud computing.

2.2 Energy Benchmarks and Metrics Assignment

Energy (measured in Joule or Wh) and power (measured in J/s or W) are the metrics, which can be used to characterise consumption of IT and ICT systems. However, they are not specific to IT. In literature, other specific measures have been proposed. We can summarise them into three broad categories:

- Power, in terms of consumed Watts.

- Efficiency, as the ratio of useful energy and total energy used.

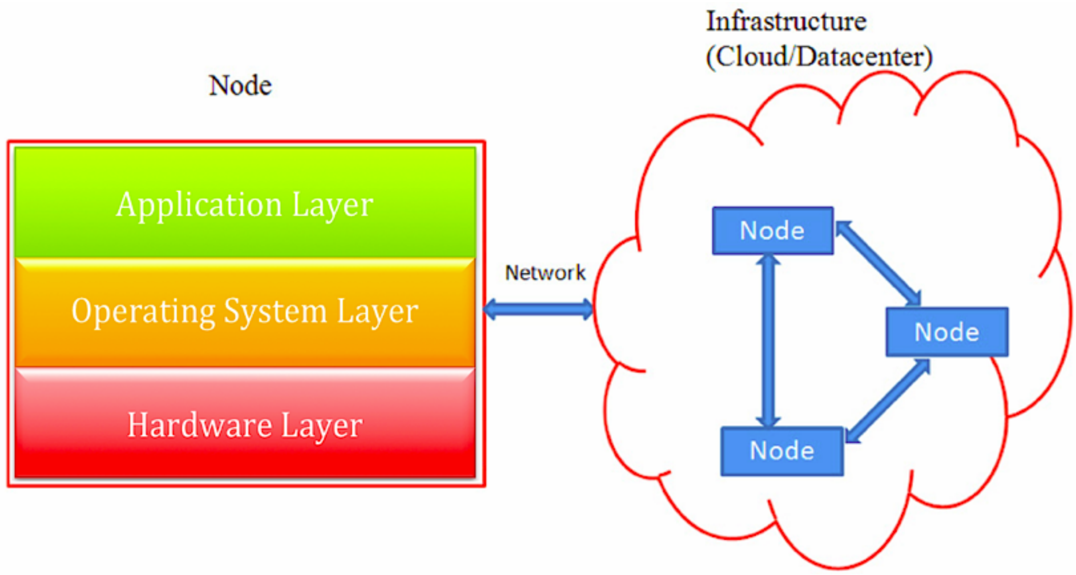

Figure 2.2. The space axis: Nodes and Infrastructure

- Productivity, defined, at high level, on a production process, as output/resource on a time interval (ex. cars produced per worker in a day). In the context of Energy and IT, the output is computational work while the resource is energy. Computational work needs to be defined at each level of the taxonomy. For instance: in a CPU, an example may be operations performed, in a network bits transmitted, in a web application hits managed.

Table 2.1 reports measures proposed in literature [84], [17] and [39] placing them in our taxonomy and categorising them as efficiency or productivity. Not all layers of the taxonomy are covered, meaning that other efficiency or productivity measures could be defined.

Table 2.1: Energy Metrics and Benchmarks

Layer	**Category**	**Unit**	**Description**	**Example**
Infr	Productivity	Useful work/W	Green Grid Data-centre Performance Efficiency (DCPE) [38]	Aims at measuring “Useful work” delivered by a data centre vs. the power used
Infr	Efficiency	%	Useful Power (for storage, computation, communication) / Total power	Green Grid Data-centre Efficiency(DCE) [38]
Node	Productivity	MFLOPS/W or FLOP/J	Number of Floating point operation computed per watt	The Green 500 list[2] ranks high performance computers on MFLOPS/W, instead of the usual ranking on FLOPS only
App	Power	W	Power used by an application on a node	Joulemeter [54] is a tool able to estimate instant power consumption of PCs and applications

[2] `http://www.green500.org/` Last Visited: 11 Apr 2013

App	Productivity	Operation / J	Output is intended as sorted records	Joule Sort [83], counts sorted records per Joule, and is a benchmark for sorting algorithms. It is a variant of Sort benchmark that considers records sorted per second
OS	Power	W	Power used by an OS or an app on a node	Softwatt [42] is a tool to estimate power used by OS and applications
HW	Power	W	SimplePower [3]	Given an instruction (or an instruction set) and a program, it estimates energy consumption on the CPU).
Network	Efficiency	% 100(MI)/M	I = energy consumption at idle, M energy at maximum	Environmental Performance Index (EPI) [67]
Network	Productivity	KB/J	KB transferred per Joule over a channel	Energy-efficiency rating (EER) [21] does the same in Gbps/W

2.3 Energy Consumption and Carbon Footprint

A lot of data has been published on IT-related consumption and emissions. However these data are usually sparse. In this section we summarise them using the taxonomy

[3] `http://www.cse.psu.edu/~mdl/software.htm` 11 Apr 2013

as a unifying point of view. In 2.3.1 we review data at a global level, then we will focus on the space and time view (2.3.2, 2.3.3), and further on.

IT is responsible for very limited percentages of consumptions and emissions as described in Figure 2.3 (less than 1% consumption of primary energy, around 3% of electrical energy, 2% of carbon emissions). However, an analysis of future trends shows that emissions and consumption tend to increase. At this level much work should be done to stabilise and possibly reduce these trends, in order to provide a sustainable IT future. The analysis performed before considers the whole of IT. In the following table, we analyse the problem considering the spatial and time dimensions, as introduced by the taxonomy.

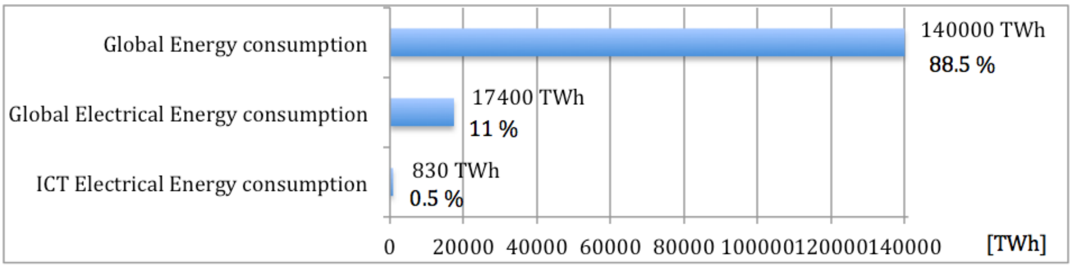

Figure 2.3. Comparison of different energy consumptions

2.3.1 Worldwide space dimension

Focusing on the IT sector, a key question is which IT components are responsible for these trends. We start from individual components (or nodes such as PCs, fixed and mobile, and data centres) and we conclude with networks.

Node view (PCs, Laptops, Mobile Devices) and Data Centres

According to [103] we report some numbers about the number of different class of devices and their consumption from 2002 to 2020. The number of worldwide PCs is expected to grow from 592 million (2002) to more than 4 billion (2020) and, regarding energy consumption, laptops will overtake desktop computers by 2020. In 2002, emissions of PCs and monitors were 200 MtCO2e, growing to 600 MtCO2e by 2020. The number of servers in 2002 was 18 million and there will be a sharp increase of this figure up to 122 million in 2020. In 2002 data centre emissions were approximately equal to 76 MtCO2e and this value should more than triple by 2020 to 259 MtCO2e. In 2002 there were 1.1 billion mobile devices. This is expected to grow to 4.8 billion in 2020. Telecommunications emissions rise from 150 MtCO2e in 2002 to about 350 MtCO2e in 2020. Figure 2.4 represents the carbon emission in

2002 [103], and Figure 2.5 reports an estimate, based on [103] of carbon emissions by IT by 2020.

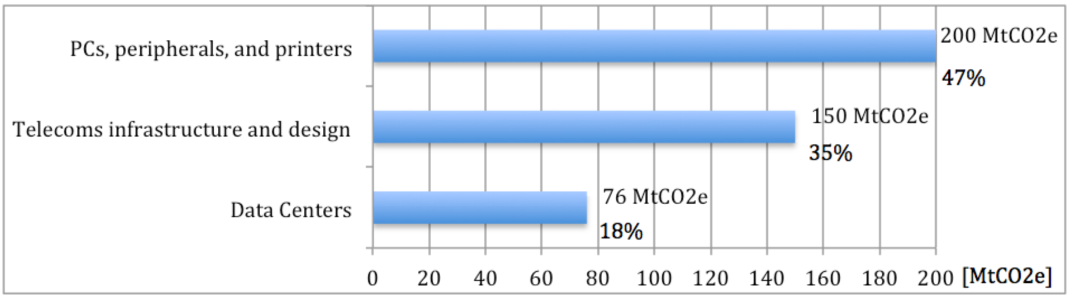

Figure 2.4. The global footprint by subsector 2002

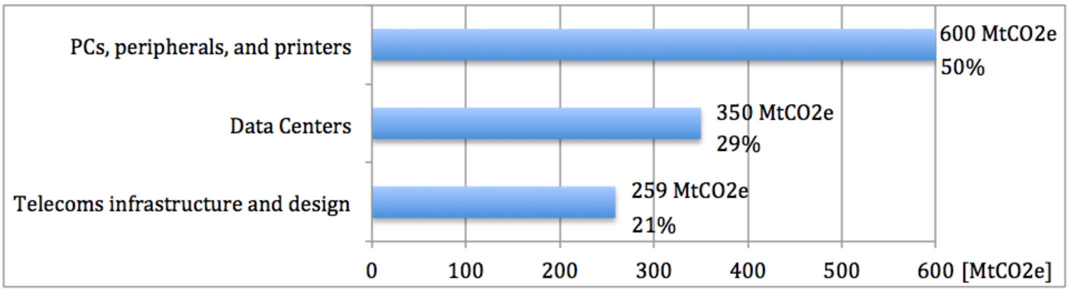

Figure 2.5. The global footprint by subsector 2020

The majority (57 percent) will come from PCs, peripherals, and printers, while data centres account for 25% only. Figure 2.6 compares the growth of devices produced and the growth of carbon emissions [103].

From the analysis above it is clear that PCs and mobile devices are the key factors in consumption in the future. Another study [48] stated that data centres in 2009 consumed about 330 TWh worldwide, and authors in [94] estimated the electrical energy consumption of PCs, laptops, and mobile phones in 2009:

- PCs: 163.2 TWh
- Laptops: 46.2 TWh
- Mobile Phones: 44.6 TWh

These data are calculated in terms of electrical energy consumed and cannot be compared with data in Figure 2.4 because of different units, years, and data aggregations.

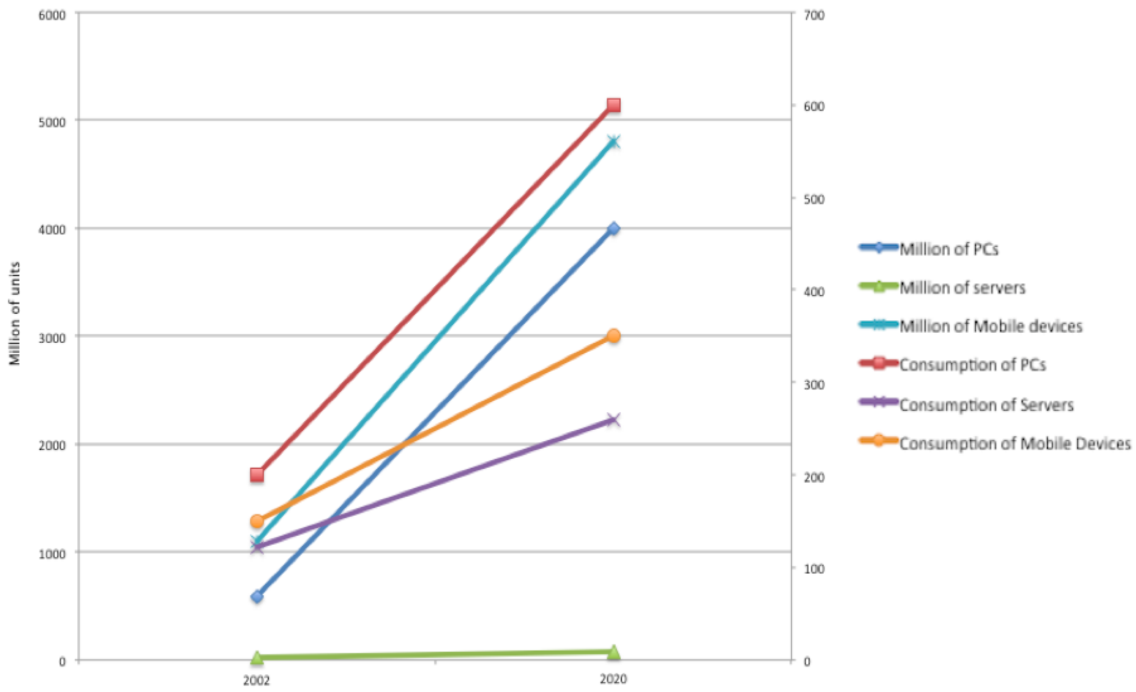

Figure 2.6. PC, Servers, and Mobile Phones production vs. Carbon Emissions

Network view

The number of Internet users has continuously increased, and the energy consumption of the Internet has grown accordingly. Network equipment such as hubs, switches and routers for the Internet consumed energy of about 6.05 TWh/year in 2010 as shown in Figure 2.7 and it is expected to grow by 1 TWh or more per year.

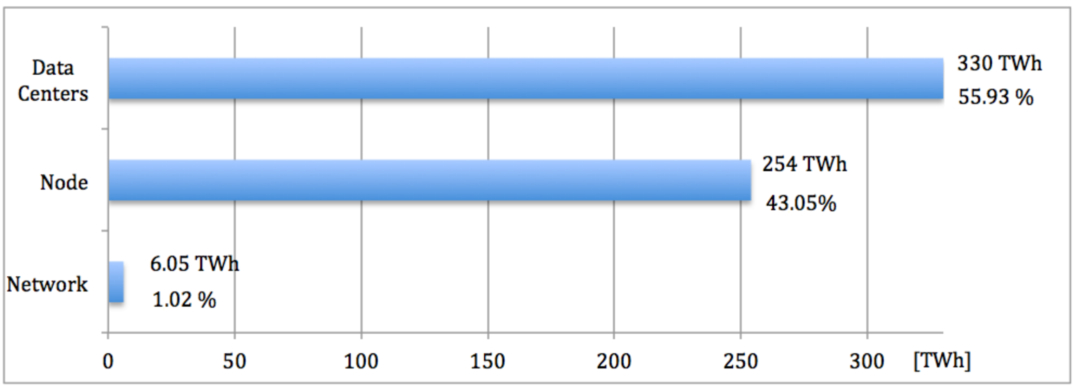

Figure 2.7. Electrical Energy consumption of IT devices based on Figure 2.2

In addition, the network equipment that connects to the Internet in home and office networks transmits packets via Ethernet links. The estimated energy consumption of Network Interface Cards (NICs) and other network devices, which use Ethernet links in the US, was approximately 5.3 TWh/yr in 2005 [77]. Moreover, the default link rate of the Ethernet and the network edge devices is rapidly increasing from 10 Mbps to 1 Gbps or more, and the number of network devices is also increasing [16]. To sum up, based upon our taxonomy, most of the energy consumption is concentrated within data centres and nodes. Network devices have a less important role but not negligible because of the magnitudes involved.

2.3.2 Local space dimension

Within a Node

In Figure 2.8 we analyse consumption inside a node [72].

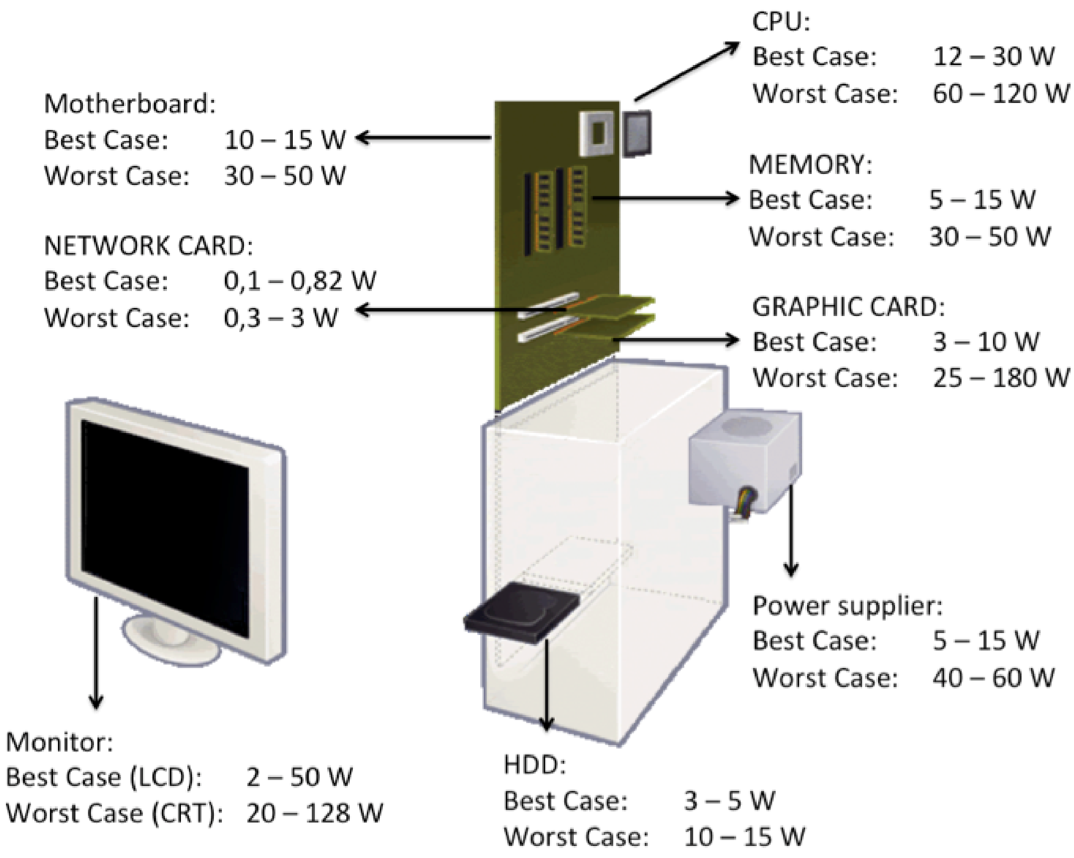

Figure 2.8. Typical energy consumption of PC components based on [72]

The energy contribution due to software can be measured on hardware. This, from the hardware perspective, is seen as a different trend of the instant power consumed by the device. There are several (and canonical) ways to gather energy consumption data from a hardware device and we cite four examples. In 1998 authors of [87] stated that the current drawn by a processor could be measured using an oscilloscope with a shunt resistor, connected in series with the supply voltage pin of the microprocessor. It is also possible to perform a direct (current) measure by utilising an ammeter with a high frequency signal to obtain a stable value [97]. In [58] the authors connected a multimeter to a laptop power supplier. They also plugged the laptop supplier into a universal power supply (UPS) to filter out voltage fluctuations. Their multimeter sampled the current 11-12 times a second. Otherwise it is possible to use smart meters; some commercially available products are Plogg Meter[4], Kill-a-Watt / Tweet-a-Watt[5], and SmartLink[6]. They aim more at monitoring, controlling, and automating the attached device. It is very difficult to find reliable values about CPU power consumptions. Usually manufacturers publish the "Thermal design power" (TDP). This value does not match the maximum CPU power consumption but it refers to the maximum amount of power that the cooling system in a computer has to dissipate and the result is expressed in Watts. These values are not comparable between CPUs produced by different manufacturers. AMD introduced a new metric (called ACP) to measure the CPU power consumption [7]. ACP is obtained by measurements taken on specially instrumented components in particular conditions (temperature, workloads, configurations). In [51] authors analysed the problem of estimating system power consumption. They stated that for each task, it is possible to measure the number of clock cycles executed per unit time and generate a model that can predict the watt consumed: P(System) Power(Taski) + P(bias') Where bias' equals every other component and Power(Taski) = F * number of FP Cycles + I * number of Int Cycles + M *number of Memory Cycles. Authors in [47] studied deep server hard drives (3.5) energy consumption, both from a mechanical and electronic perspective. Based on their studies, drive platters spin constantly, they stop only in standby mode; read/write heads are only powered during the reading and writing phase. The arm actuator is only powered when there is the need to seek across locations on a platter. Printed circuit board electronics are instead always powered. Based on ATA/ATAPI-5 specification and the Advanced Configuration and Power Interface (ACPI), modern hard drives support four power management states: active, idle, standby, and sleep (it is not possible to recover from this state without a system reset). In standby mode, mechanical energy savings may

[4] `http://www.bytesnap.co.uk/bytesnap-design-case-studies-the-plogg-smart-energy-meter/` Last Visited: 11 Apr 2013

[5] `http://www.ladyada.net/make/tweetawatt/` Last Visited: 11 Apr 2013

[6] `http://www.smarthome.com` Last Visited: 11 Apr 2013

range from 92 % to 99% and electronic components energy saving may span from 35% to 95%. In idle mode mechanical components can consume from 25% to 75% of the total energy consumption. During the read/write phase the energy consumption is dependent on the Logical Block Number (LBN). Data density increases at higher LBNs and more time is required to recover the data (due to constant angular velocity of the spindle), read bandwidths decreases and read energy consumption increases. But write bandwidths remains fairly constant because write requests are not influenced by the varying data density on the platters, hence bandwidth and energy consumption do not vary on the basis of LBN. In general, reading consumes more energy than writing for blocks larger than 2KB, and writing is more energy intensive than reading for blocks smaller than 2KB.

Disk Model Unit	**Cap.** GB	**Read** MiB/s	**Write** MiB/s	**Read** W	**Write** W	**Idle** W	**Efficiency** MiB/j
SSD							
Intel X-25E	32	226	198	1.7	2.7	0.6	103
Intel X-25M	80	225	79	1.0	2.5	0.6	128
Samsung PB22-J	256	201	180	1.1	2.8	0.6	124
Super Talent FfM56GX25H	256	235	163	1.6	2.9	0.5	102
HDD							
Samsung HD502HI	500	106	108	6.6	6.6	3.7	16
Samsung HM500JI	500	87	87	2.3	2.3	/	28
W.D. WD7500KEV	750	82	82	2.0	2.0	/	41

Table 2.2: Energy Metrics and Benchmarks

The energy consumed during seeking is minimal, but restricting disk accesses between low and central LBNs could help to save energy in long usage periods. Solid-state drives have different power consumption profiles and efficiency as shown in Table 2.2. Authors in [86] analysed energy consumption of office and telecommunications equipment in commercial buildings. They divided monitors and printers into categories. Table 2.3 and Table 2.4 show the energy consumption of monitors divided into two categories: CRT and LCD. Over the years monitors show an increase of performance and a decrease of energy consumption.

Table 2.5 shows the energy consumption of different categories of printers.

Type - CRT	Active	Standby	Suspend	Off
14-15”	61	53	19	3
17”	90	26	9.2	4.3
19”	104	31	13	4
21”	135	43	14	4.7

Table 2.3: CRT energy consumption estimate [86]

Type - LCD	Active	Standby	Suspend	Off
13”	2.5	0.7	0.2	0.1
14”	6.7	1.9	0.7	0.3
15”	11.7	3.4	1.2	0.6
17”	16.7	4.8	1.7	0.8
18”	25	7.2	2.5	1.2
20”	31.7	9.2	3.2	1.6
21”	36	10.4	3.6	1.8

Table 2.4: LCD energy consumption estimate [86]

Within Network, Data Centres and Infrastructure

Authors in [41] examined the energy consumption of networking devices in the Internet, based upon data collected by the U.S. Department of Commerce. Data are available in Table 2.6.

Based upon these findings, the impact given by a reduction in consumption in this field can be relevant.

2.3.3 Time dimension

PC

Let us now analyse how these consumptions are distributed over the lifecycle of IT devices (manufacturing and transport, usage, disposal). According to [105] the energy to manufacture a PC accounts to 4250 MJ, the energy spent in usage (considering an average usage time of 3 years) is 1500 MJ, and the overall energetic cost (including transport and purchase) is about 7900 MJ. Manufacturing is the most energy-hungry phase, and consequently the author suggests concentrating efforts on reusing devices to extend their average usage time.

In detail the distribution of energy consumption [73] (Figure 2.9) is as follows:

- Design/Manufacture: 4250 MJ (54%)

Device	Active	Standby	Suspend	Off
Impact Printer	36.5	16.8	N/A	1
Inkjet Printer	42.576	13.377	N/A	2.878
Laser Printer	231	28	16	1.9
Laser Printer Small Desktop	130	75	10	N/A
Laser Printer Desktop	215	100	35	N/A
Laser Printer Small Office	320	160	70	N/A
Laser Printer Large Office	550	275	125	N/A

Table 2.5: Printer energy consumption estimate [86]

Device	Approximate Number Deployed	Total AEC TW-h
Hubs	93.5 Million	1.6 TW-h
LAN Switch	95000	3.2 TW-h
WAN Switch	50000	0.15 TW-h
Router	3257	1.1 TW-h
Total		6.05 TW-h

Table 2.6: Energy consumption of networking devices in the Internet [41]

- Transport: about 950 MJ (12%)
- Purchase/Use: about 1500 MJ (19%)
- Other 1200 MJ (15%)

According to [73] the energy taken for the production of a common Personal

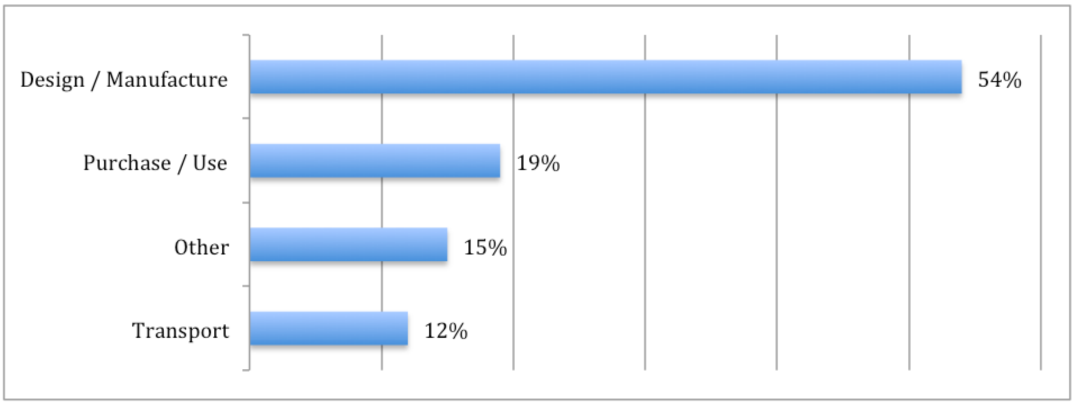

Figure 2.9. Phases and Energy Costs

Computer has risen by 7% in comparison to 2002. This value increased from about 6420 MJ in 2002 to about 6900 MJ in 2007. About the usage phase, modern PCs consume more energy at full-load than the old ones, while in a low-power mode they take less energy than the old computers. Therefore the total energy used depends strongly on the usage scenario. Considering a home scenario, the total energy used by an average 2007 PC per year is almost the same as it was in 2002. The reduction is mainly due to CRT monitors (between 65 W-145 W when active, and 9-14 W in standby) substituted by LCD monitors (25 W when active, 2 W in standby). The increase is due to a possible increase in consumption of other components of a personal computer (graphics cards, memories, etc.) Thus, the overall (manufacturing + usage) energy consumption of PCs has increased over the last 10 years. According to the Wikipedia definition [7], despite not being an official source, we can draw a trend about the frequency and TDP behaviour of AMD CPUs since 1996 as described in Figure 2.10.

Without taking into account punctual values we can see that over these 15 years both frequency and TDP raised respectively 7 and 3 times. We must point out that nowadays CPUs have multiple cores so TDP and Power Consumption are not the right metrics to measure CPUs energy efficiency.

Data Center, and Infrastructure

According to [15], the major challenge in energy reduction talking about "Cloud Computing" is the relation among system components and an optimal balance between performance, QoS, energy consumption, and self-aware runtime adaptation.

[7] `http://en.wikipedia.org/wiki/List_of_CPU_power_dissipation` Last Visited: 11 Apr 2013

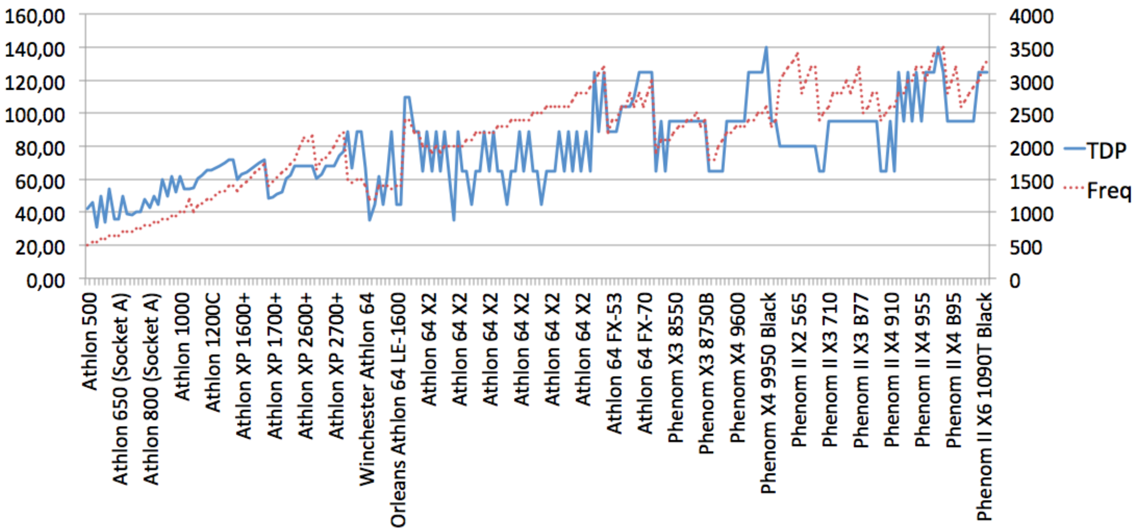

Figure 2.10. CPU Frequency and TDP trend

Amazon [44] calculated that the cost and operation of servers amount to 53% of their total budget, while energy-related costs reach 42% of the total. Figure 15 shows a typical monthly cost distribution in a data centre.

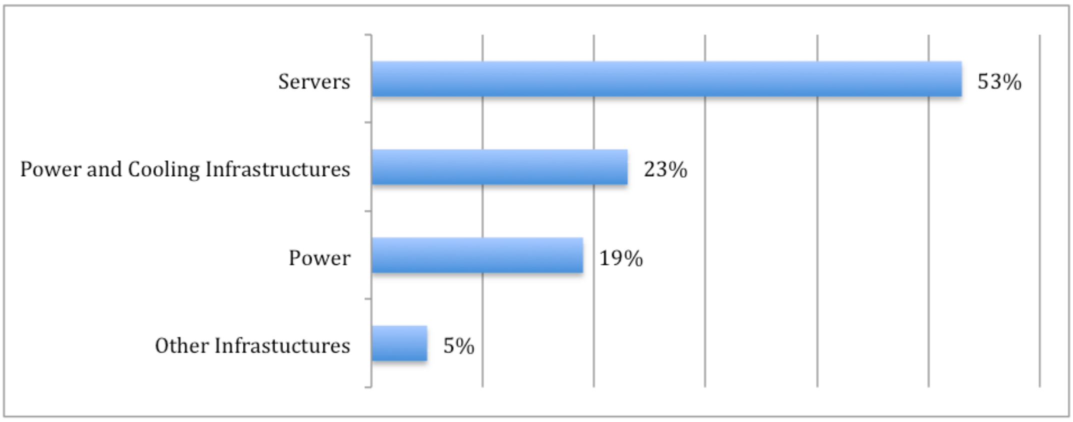

Figure 2.11. Typical data centre monthly costs distribution [15]

According to [57], the electricity used in global data centers in 2010 accounted for between 1.1% and 1.5% of total electricity use. This means that data centers are using less energy than predicted by Environmental Protection Agency in 2007.

2.3.4 Considerations

As a conclusion, an analysis of the trends in IT energy consumption and emissions shows that the IT sector is responsible for a minority of them. However, the trend is increasing, due to the large increase in the number of individual IT devices (PCs and mobile phones), and the increase in energy used to manufacture them and, in using them. It is a responsibility of the IT sector to research ways to become sustainable, and to further reduce emissions and consumption, especially at the level of individual devices.

2.4 Energy Efficiency: guidelines

As shown in Section 2.2, the literature proposes many tools to measure Energy Efficiency of IT devices and data centres. There are many works aimed at reducing the consumption of PCs in enterprise environments [5] [6], or to manage the use of green energy in datacentres [36] [34] [35] [90] [92], or to optimize the consumption of idle server [70]. The next point is how to obtain Energy Efficiency also by software, thus considering the OS and application layers and not only the hardware layer. We have surveyed the literature and we have found guidelines from Intel and Siemens [53] [95] [61] and also from Academy [40]. Most of the guidelines suggested in the literature are not strictly code-related, but they are mainly high-level recommendations for programmers and software designers (e.g. to implement lazy loading of libraries). However, it is worth mentioning that such guidelines, despite being intuitive and acknowledged as effective by software industry specialists, did not receive any empirical validation. For this reason, an empirical validation that quantitatively assesses their impact on Energy Efficiency is needed. We summarise these guidelines below, according to the taxonomy proposed in Section 2.1.

2.4.1 Infrastructure

1. **Consider Cloud Platforms for energy-efficient Internet applications**: Cloud platforms use virtualization. This should improve energy efficiency of the internet application [61].
   Pros:

   - Reduced costs.
   - Hardware reduction.
   - Less Power Consumption.

   Cons:

- Single point of failure.
- Lower performances.

2. **Load balancing**: Distributing workload evenly across two or more computers, network links, CPUs, hard drives, or other resources.[40] reduces the CPU temperature
Pros:

   - Reducing the use of a mobile device can increase its average battery lifetime.

   Cons:

   - Load balancing can be difficult to accomplish.

3. **Provide information for system management tools to support overall optimisation**: The use of power meters and energy-aware applications provides information relating to the infrastructure energy consumption. This data should be used as an input to support the energy optimization [53].
Pros:

   - Instant Power consumption is available as a Context Information [12].

   Cons:

   - This solution requires dedicated hardware.
   - The instant power consumption information must flow from the hardware layer through the application layer. It is necessary that all the layers are prepared to handle this information.

2.4.2 Application

1. **Design Efficient UI**: Efficient UIs can let the user complete a task quickly. An inefficient UI can increase the application complexity, and consequently the energy consumption [40].
Pros:

   - An efficient UI may improve the energy consumption of the entire application by simplifying the interaction with the user, and reducing the time required to perform a specific operation

   Cons:

   - It requires a specific study of the UI.

- It requires an empirical validation of the changes.

2. **Use Event-Based Programming**: Event-based programming avoids a waste of resources involved in doing unnecessary operations. If polling cannot be avoided, it is advised to select a fair time interval [40] [53] [61].
   Pros:

   - The application is put to sleep when not used.

   Cons:

   - The application may need to be redesigned.
   - It is not always implementable.

3. **Use low-level programming and avoid use of byte-code**: With low-level programming languages, developers have more details of the system which she/he is developing, than when using high-level programming languages. When possible, it is advised to develop the more computationally intensive parts of the application at a low-level. Programming languages to increase performances and energy efficiency. The virtual machine interpretation of byte code can make the application energy inefficient [40] [61].
   Pros:

   - The developer has a greater control of the underlying hardware by developing at a lower level of abstraction.

   Cons:

   - Can be expensive in terms of costs and development time.
   - Not always possible.

4. **Batch I/O**: Buffering I/O operations increases energy efficiency; the OS can power down I/O devices when not used.[40] [53] [61]
   Pros:

   - Known technique already used for generic optimisations.

   Cons:

   - Not always possible. It depends on the application context.

5. **Code Migration**: To increase energy efficiency in devices where computation can be energy consuming, it may be worth moving the task to another energy-efficient environment and gathering results when available.[40]
   Pros:

- It simplifies the “client-side” code.

Cons:

- It is needed to consider the overheads due to the use of another system, which performs complex operations.

6. **Reduce data redundancy**: Storage and transportation of redundant data lowers energy efficiency.[40]
Pros:

   - To be applied during the design phase.

   Cons:

   - A later change to an existing implementation can cause the modification of a large part of the application.

7. **Reduce QoS/Scale dynamically**: The application has to be able to change its behaviour in case of low-power situations.[40] [53] [61]
Pros:

   - The creation of user profiles can save energy [12]

   Cons:

   - It needs metrics.
   - The QoS reduction is not always possible.

8. **Use Power/energy profiling tools**: This kind of application models the energy behaviour of the system in which it is run. This model should help the developer to optimise the application energy usage [53].
Pros:

   - With these tools it is possible to create energy consumption models, which help to predict the energy consumption in different situations.

   Cons:

   - Models change based upon the device.

2.4.3 Operating System

1. **Implement Power Management APIs**: The Operating System can export power management APIs to enable applications to manage energy efficiency at lower levels.[53] [61]
   Pros:
   - APIs are easily used by the application layer.

   Cons:
   - Needs OS modification.

2. **Optimal use peripherals**: The use of a correct power management feature exploits properties and characteristic of peripherals[40]
   Pros:
   - The OS can use the peripherals in different operating modes.

   Cons:
   - Manufacturers have to create drivers that allow the OS to use the device in different operating modes.

3. **Use Compiler Optimisation**: The Compiler can optimise the source code according to specific platform architecture.[61] [40]
   Pros:
   - Low developing costs.

   Cons:
   - Not all the architectures can get the same optimisations.

4. **Use only required services and background processes**: Unused applications waste memory, resources and energy. [61]
   Pros:
   - It is possible to free resources used by unnecessary idle processes.

   Cons:
   - Killing a process can cause a device to misbehave.

2.4.4 Hardware

1. **Power down peripherals**: Peripherals can be storage, I/O, or network devices, GPS modules, etc. When not in use they should be set to a low power state or shut down. [40]
   Pros:
   - The upper layers improve their performance in terms of energy consumption transparently.

   Cons:
   - It is needed to limit the overheads caused by the transition between ON and OFF.

2. **Use specific-purpose hardware**: A general-purpose hardware can be oversized for the specific problem. Oversized hardware can be translated in energy inefficiency. [53]
   Pros:
   - Maximum optimisation of energy consumption for the current device.

   Cons:
   - High costs.

3. **Dynamic Power Management Capabilities**: Exploit features such as ACPI, processor and idle management, configurable high performance vs. low power components, to manage energy consumption. [53]
   Pros:
   - It provides to higher layers different configurations to save energy.

   Cons:
   - It is needed to redesign devices.

4. **Power/ Energy metering support**: Hardware metering support is often needed by profiling tools to get a reliable estimation of device power consumption. [53]
   Pros:
   - Provides information about the instant energy consumption of the device to the upper layers.
   - It is possible to create more accurate models.

Cons:

- High costs.

2.4.5 Network

1. **Efficient data traffic**: Sending less data over the network can reduce energy consumption because the network interface is left in idle for more time [40].
   Pros:

   - This action can also improve the application performances.

   Cons:

   - In order to send less traffic, the application architecture could be deeply modified.
   - It is necessary to take into account the additional computation needed to implement data compression, proxying, etc.

2. **Energy impact of communication protocols**: An energy-efficient communication protocol can reduce the amount of information exchanged.[53]
   Pros:

   - If a protocol is designed to reduce the amount of data exchanged to establish a communication, its correct implementation can give advantages in terms of energy consumption.

   Cons:

   - It is needed to modify/extend the existing protocols.

2.5 Considerations

Green IT is becoming a popular topic, but no specific surveys are available yet. In this paper we reported a survey of the literature about Energy Consumption & IT systems, starting from the viewpoint of Green IT. The survey has been performed searching for the following keywords: "Green IT", "ICT Energy Consumption Reduction", "Energy Efficiency", "Energy Measurement", "Power management", "Energy Consumption Analysis" in the following databases: IeeeXplore, ACM digital library, IET Electronic Library and, more generally, Google Scholar. In order to organise the large number of papers found we have defined a taxonomy, based on two axes, the time axis (with activities such as design, manufacture, transport, use, dismiss and possibly recycle) and the space axis (with physical components of varying sizes, from larger to smaller: the clouds, data centers, computing nodes such as PCs, smartphones and mobile phones, applications, OS and hardware). In 2007 IT electrical energy consumption in the usage phase is reported to be 830 TWh or 0,5 % of the total. In percentage this is a minimal amount; however, in absolute terms it is relevant. Besides, estimates of IT consumption in the future show a fast-growing trend. While we do believe that these figures are a good starting point, it should be noted that the accuracy of data reported is questionable. For sure consumption data is in many cases referred to several years ago (2007, 2009 for IT consumption) and their precision is not reported. As regards to this, a lot of work should be done to define and standardise the way consumption data is collected and reported. A first observation on the space axis is that there is no agreement on how to consider smart phones and mobile phones in general. Sometimes papers do not include them (strict IT and Green IT), sometimes they do (ICT and Green ICT), and sometimes the point remains fuzzy. Overall our point of view is that, considering the convergence of mobile phones into Internet nodes, they should be included. Besides, nowadays the production (and therefore the related consumption) of mobile phones is much larger than that one of computers. Most of the literature is about the space axis, and mainly about the usage phase. However, considering PCs at least, a study [72] shows that the main contribution (about 50%) of energy consumption of a PC is due to the design/manufacture phase while the usage phase contribution represents only 20% of the total. The energy consumption of the usage phase, which currently does not reach 1% of the total, can be considered the one in which it is possible to intervene in a more distributed way. Small and distributed energy reductions can lead to large reductions worldwide. For this reason, an intervention on energy consumption reduction from a software point of view is to be considered interesting. We did not find similar studies on mobile phones or data centres. However, if the trend is confirmed, efforts to reduce energy consumption should concentrate on the manufacturing phase and/or on increasing the duration of the usage phase. Again considering the space axis, in 2009 data centres are the main users of energy (330 TWh) followed by (PCs, smartphones, tablets) (254 TWh) and network equipment (6TWh). From these figures is clear that data centers and PCs/smartphones should be the focus for energy consumption reduction during the usage phase. After this data collection and analysis phase, we have focused on methods and techniques to reduce energy consumption. In this regard we need precise ways to measure if consumption is actually reduced. So before all we have summarised the measures that can be used. Besides the obvious ones (energy and power) we have surveyed what has been proposed, and placed it into our taxonomy. Basically all measures proposed by different authors can be classified as measures of efficiency or productivity, applied to a node of the taxonomy in the usage phase. For instance efficiency for a data centre is the ratio between energy for computation and total energy used (including conditioning). Finally, we have surveyed for techniques (or guidelines) to reduce consumption, and organized them in our taxonomy. What is available is a good starting point, but in many cases the guidelines are quite high level, so their effect on consumption is hard to express in quantitative terms. In summary, future work by the Green IT community should be devoted to:

- Collect more precisely data about consumption, standardising the data collection process;
- Collect and analyse in more depth consumption in the non-usage phase;
- Develop more extended and more detailed guidelines for energy consumption reduction;
- Validate and characterise quantitatively the effect of these guidelines.

The next chapter discusses Energy Consumption Measures and it will also investigate whether Energy Efficiency is eligible to be included in a software quality model.

Chapter 3

Energy consumption and quality models

3.1 Energy Consumption Measures and Profiling

This Chapter deals with RG1 and RG2 by analysing how to measure the energy consumption of a software application, and proposing to consider energy efficiency as a software non-functional requirement. As described in Section 2.1.2, energy (measured in Joule or Wh) and power (measured in J/s or W) are the metrics, which can be used to characterise consumption of IT and ICT systems. However, they are not specific to IT. In literature, other specific measures have been proposed. We can summarise them into three broad categories:

- Power, in terms of consumed Watts.
- Efficiency, as the ratio of useful energy and total energy used
- Productivity, defined, at high level, on a production process, as output/resource on a time interval (ex. cars produced per worker in a day). In the context of Energy and IT, the output is computational work while the resource is energy. Computational work needs to be defined at each level of the taxonomy. For instance: in a CPU, an example may be operations performed, in a network bits transmitted, in a web application hits managed.

More details are introduced in Table 2.1.

In any programmable device, although the ultimate responsibility of energy consumption is always with the hardware, the way the energy is consumed is dictated by the software. Consequently, it is necessary to draw a theoretical model of the energy consumption, which depends both on hardware specifications and the way in which they are used by software artifacts. The *abstract* model underlying the power consumption can be summarised as:

$$Power = Idle + \sum_{c \in Components} Hw_c \cdot Sw_c$$

The total power consumption $Power$ of an IT device – when turned on – is composed by an $Idle$ part that is present even when the device is sitting idle. The additional consumption depends on the individual hardware components maximum consumption Hw_c which is modulated by what the software forces it to do, Sw_c. Depending on the software requests the hardware component may run at full throttle or remain idle.

This theoretical software power model supports higher – software level – strategies for increasing software energy efficiency. As a matter of fact, it gives developers a way to elaborate a strategy by analysing the causes of energy consumption and also to validate the efficacy of the formulated strategies by measuring their impact and effect. In this chapter, a framework for energy-efficient software strategies will be introduced. The framework is represented in Figure 3.1.

The Refactoring strategy is shown on the left side. It focuses on minimizing software instructions and code patterns (Energy Code Smells) that may cause higher energy usage. This is achievable at design time, following code level guidelines, and at implementation time, detecting the Energy Code Smells. The Self-Adaptation strategy, shown on the right side of the figure, has the main goal of creating an energy-aware application that is able to choose among different configurations, with respect to different scenarios and contexts, that we call “Energy Profiles” (see Section 4.5). The two strategies are not meant to be mutually exclusive: they can be applied together in the same development process. In addition, other technological, human or process strategies can be plugged in whenever their impact is measurable and linkable to a software application and its power consumption, through modelling and profiling. Both strategies should be applied iteratively, by verifying the energy efficiency improvements, through power profiling tools, and consequently adapted. They also should be applied carefully, keeping in mind the software mission and its main functionalities, the required quality of service, and the interests of the stakeholders. For example, applying Self-Adaptation to reduce the network usage might improve energy efficiency, but it might also violate Service Level Agreements on response time or availability. Therefore, the stakeholders’ network around the software should be drawn in order to understand who might be affected by adopting such strategies.

The bottom part of the figure shows the “Response” level, meant to identify opportunities for energy optimisation and/or to assess the energy savings gained by applying our strategies. Resource usage information, such as memory accesses,

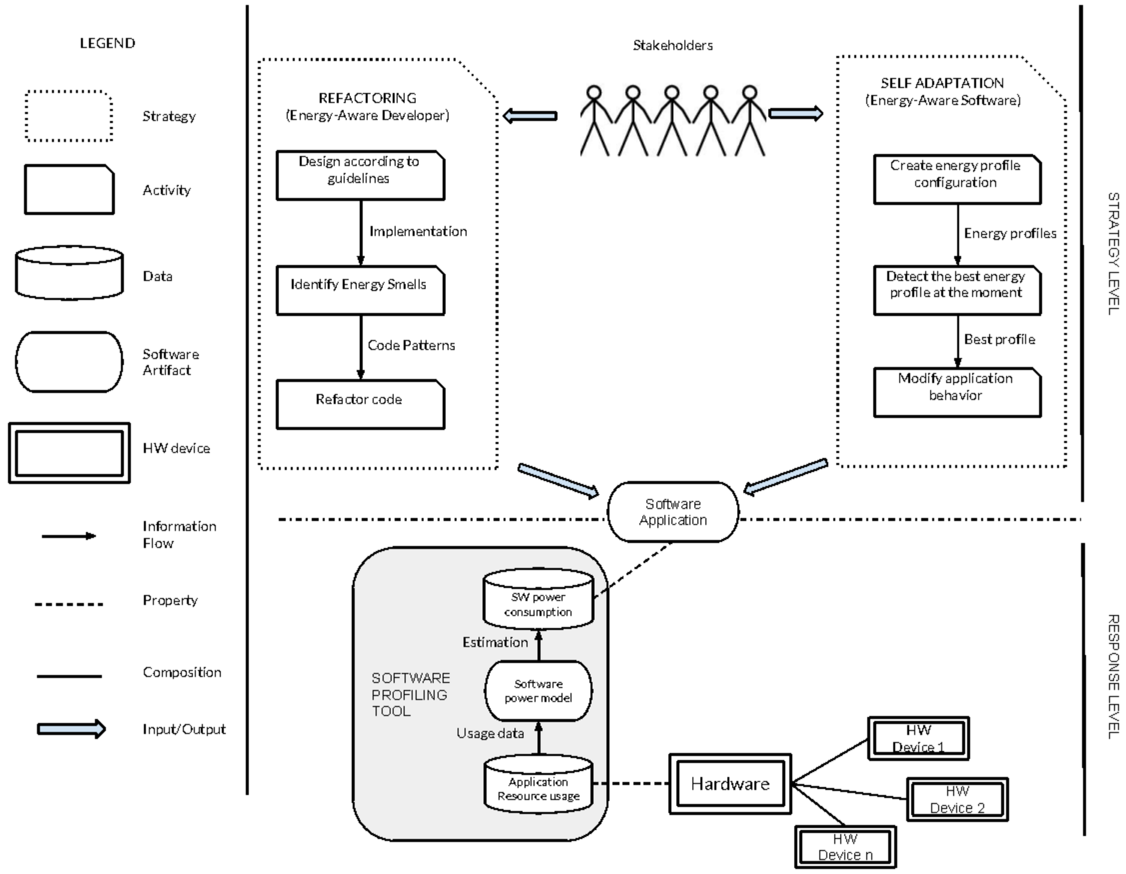

Figure 3.1. Framework for Energy-Efficient Software Strategies

device usage, CPU mode and such, is collected from the hardware. This information is used as input for software profiling tools, that analyse software applications during execution and provide on-line power consumption estimation values with different granularity. Software power models represent a crucial component of our framework, because they enable the formulation and validation of the strategies. The software power consumption models are described in more detail in the following section, along with their implementation inside software energy profiling tools (see Section 3.2: Software Power Profiling Tools).

3.2 Software Power Profiling Tools

This section will introduce some useful tools that developers may use in order to profile the power consumption of their application. According to the framework presented in Figure 3.1 these tools are included in the "Software Profiling Tools" block; the developer, after implementing one (or both) strategy (Refactoring and/or Self-Adaptation) can verify the energy efficiency improvements, through such power profiling tools.

3.2.1 Joulemeter

Developed by Microsoft Corp.[1], Joulemeter is a software tool available for Windows platforms, which provides a power consumption estimation for PCs. It needs an initial calibration phase, which is done using battery data (on laptops) or using an external power meter (on desktops). Currently, only one meter is supported: the WattsUp Pro meter. There is also an option for manually inserting calibration values, namely idle power consumption, peak CPU consumption (for high and low frequencies) and monitor consumption. After calibration, Joulemeter provides power consumption estimation in real time, with components breakdown (CPU, monitor, disk). It is also possible to specify an application to get the estimation of a single application. The estimation error is within 3-5% of the actual value [54].

- Pros:
  - Ease of use
  - Per-application estimation
  - Good accuracy
- Cons:
  - Needs initial calibration
  - Only supports a specific model of power meter
  - Windows-only

3.2.2 ARO

Application Resource Optimizer (ARO) is a diagnostic tool for analysing mobile web applications performance, developed by AT&T. It is composed of two modules: the Data Collector and the Data Analyzer. With the Data collector it is possible to log data from a device: the collector must be started and then the target application is run. It is also possible to record a video in order to better understand the consequences caused by the user activity. The Data Analyzer processes the

[1] http://research.microsoft.com/en-us/projects/joulemeter/

data acquired by the Data Collector and provides the following features: Visibility into radio resource and energy utilisation, benchmarking of resource efficiencies, automatic diagnosis of application inefficiencies

- Pros:
  - easy to use
  - provides focused hints to improve the energy efficiency of the analysed application
- Cons:
  - works only on rooted android devices and windows 8 devices
  - does not provide real-time monitoring

3.2.3 Power TOP

PowerTOP[2] is a tool designed to monitor software power consumption in Linux-based systemsby Intel Corp. in 2007PowerTOP mainly focuses on monitoring CPU states and showing to the user which software processes are responsible for setting the CPU into high power states. In this way, the tool helps in diagnosing power consumption issues and identifying energy-inefficient software. More recent versions of PowerTOP include monitoring of the activity of devices and peripherals (e.g., USB storage, WiFi modules, GPU) and support for power-management configuration. PowerTOP is mainly designed for battery-powered devices, where it requires a calibration phase to analyse battery discharge behavior. It uses the ACPI interface to collect information on how much power is effectively consumed, and it also provides an estimation of how many hours of battery are left.

- Pros:
  - Per-application estimation
  - Power management support
  - Provides information about devices and peripherals
- Cons:
  - Needs initial calibration
  - Effective power consumption estimation only available on battery-powered devices
  - Linux-only

[2] `https://01.org/powertop/`

3.2.4 Intel Energy Checker

Intel Energy Checker[3] is an SDK developed by Intel Corp., designed for application developers who want to build energy-efficient or energy-aware software. It is available for the most common platforms The SDK provides a simple API, in C language, that developers can use to instrument their code. The instrumentation is done by adding counters that can help to identify the "useful work" done by the application: these counters need to be defined according to the specific application domain (e.g. a mail application can record the number of received emails)The second step is to interface the application with energy meters, in order to collect energy consumption information. The SDK provides drivers and libraries to communicate with different models of hardware energy meters. Then, it is possible for the developer to visualise and correlate the usage metrics with power consumption data in order to analyse the energy efficiency of its application, or to implement energy-aware policies. The SDK also provides an ecosystem of tools and utilities that assist developers in embedding the components of Intel Energy Checker into their applications.

- Pros:
  - Multi-platform compatibility
  - Fully customisable productivity metrics
  - Supports different models of power meters
- Cons:
  - Configuration of devices and components can be time-consuming
  - Low-level APIs (Wrapping of C libraries might be required)
  - Only relies on external power meters for power consumption monitoring (no estimation)

3.2.5 PowerTutor

PowerTutor is a mobile application for Android devices, which displays the power consumed by their major hardware components, including CPU, displays as well as Wi-Fi, Audio, GPS, Sensors and cellular interfaces within 5% of actual values. It builds power consumption models by controlling the device power management. PowerTutor includes a set of parameters for each state describing how the power is consumed. In Android each app/process is considered as a separate user with

[3] `http://software.intel.com/en-us/articles/intel-energy-checker-sdk`

its own UID. Under /proc/uid_stat/<UID>/ directory, a lot of information are available about the app/process, including data transmitted, memory usage etc. PowerTutor maps process id by using UIDs then, based on the stats found under the <UID> directory, it decides the states of each component. After that, based on the power consumption model, PowerTutor computes the energy consumption.

- Pros:
  - Available in the Google Play Store
  - No root privileges required
  - Graphs and charts are available or log files can be downloaded
- Cons:
  - Valid values for a subset of Android phones only
  - No APIs available

3.3 Energy as a non-functional requirement

The rapid growth and significant development of Information Technology (IT) systems has started to cause an increase of worldwide energy consumption [103]. This issue moved technology producers, information systems managers, and researchers to deal with energy consumption reduction [15]. For this reason, research has increasingly focused on improving the Energy Efficiency of hardware, but the literature still lacks in quantifying accurately the energy impact of software. Software does not consume energy directly; however, it has a direct influence on the energy consumption of the hardware underneath. As a matter of fact applications and operating systems indicate how the information is processed and, consequently, drive the hardware behaviour. Considering each IT device, it has its own theoretical energy consumption, which can range from 0, when it is turned off, to x if all its internal components are used simultaneously. Through the management of each part there is a variation Δx of its consumption that is between 0 and x. The management of system components can be done either in hardware or software. Previous work described in Chapter 4 suggested that software can reach up to 10% of the total system power (measured as the difference between an idle activity, used as a baseline, and the most power-consuming scenario). These figures ought to be taken into account, especially when considering mobile environments and data centres. Mobile handset sales are increasing sharply [3] and this class of devices has to deal with battery-related issues, so energy savings can impact significantly on the device autonomy. On the other hand, small energy reductions in data centres can result in big energy savings: for example, in 2009 alone, data centres consumed around 330 TWh [89].

3.3.1 SQALE

SQALE [62] is a methodology to support the evaluation of the software quality. It is applicable to any software artifact (such as code, UML models, documentation, and so on); however, the main focus is on source code, whose quality is perceived as a non-functional requirement. The goal of using SQALE is to quantitatively assess the distance between the code's current quality and its expected quality objective. To achieve that, the following main concepts are introduced:

1. A quality model
2. An analysis model
3. Indexes and Indicators

Quality model

The quality model proposed by SQALE is an orthogonal quality model derived from the ISO/IEC 9126 [49] (revised by the ISO/IEC 25010 [50]) . It is organised into three hierarchical levels, which are represented in Fig. 3.2. The first level is composed of characteristics that are based on the theoretical lifecycle of a source file and are from the ISO 9126 standard. They depend on the code's internal properties and directly impact the typical activities of a software application's lifecycle. Characteristics are listed in the order they appear in a typical software application lifecycle: Testability, Reliability, Changeability, Efficiency, Security, Maintainability, Portability, Reusability.

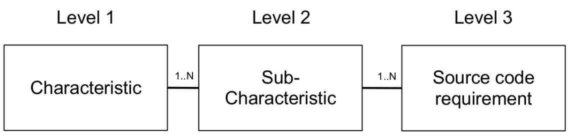

Figure 3.2. Hierarchical quality model structure

The second level is composed of sub-characteristics, based on sub-activities and requirements domain. There are two types of sub-characteristics: those corresponding to lifecycle activities (e.g., unit test, integration test), and those resulting from taxonomies in terms of good and bad practices relating to the software's architecture and coding. A sub-characteristic is attached to only one characteristic, the first in the chronology of the characteristics (to preserve orthogonality). The third level is composed of requirements that relate to the source code's internal attributes. These requirements usually depend on the software's context and language, and they are also attached to the lowest possible level, i.e. in relation to the first quality characteristic to which it chronologically contributes. In this way orthogonality is preserved at the bottom level as well. Requirements relate to the artifacts that compose the software's source code, e.g. software applications, components, files, classes, and so forth. TABLE 3.1 is an excerpt from the SQALE standard [62] and it contains examples of how requirements in the Java language are inserted in the structure of characteristics and sub-characteristics. Fig. 3.3 represents graphically the hierarchy.

Characteristic	**Sub-characteristic**	**Generic Requirement Description**
Maintainability	Understandability	File comment ratio > 35%
Maintainability	Readability	File size (LOC) < 1000
Maintainability	Readability	No commented-out code
Efficiency	RAM-related efficiency	Class depth of inheritance (DIT) < 8
Efficiency	RAM-related efficiency	No unused variables, parameter or constant in code
Changeability	Logic-related changeability	If, else, for, while structures are bound by scope
Reliability	Fault tolerance	Switch statements must have a default condition
Reliability	Data-related reliability	No use of unitialised variables
Testability	Integration level testability	Coupling between objects (CBO) < 7
Testability	Unit Testing testability	No duplicate part over 100 token
Testability	Unit Testing testability	Number of parameters in a module call (NOP) < 6

Table 3.1: Example of SQALE model for Java language, from [62]

Analysis model

The SQALE Analysis Model contains the rules to normalise and control measures relating to the code. For each violated source code requirement, a remediation cost (a work unit, a monetary unit, or a time unit) is associated to make the code

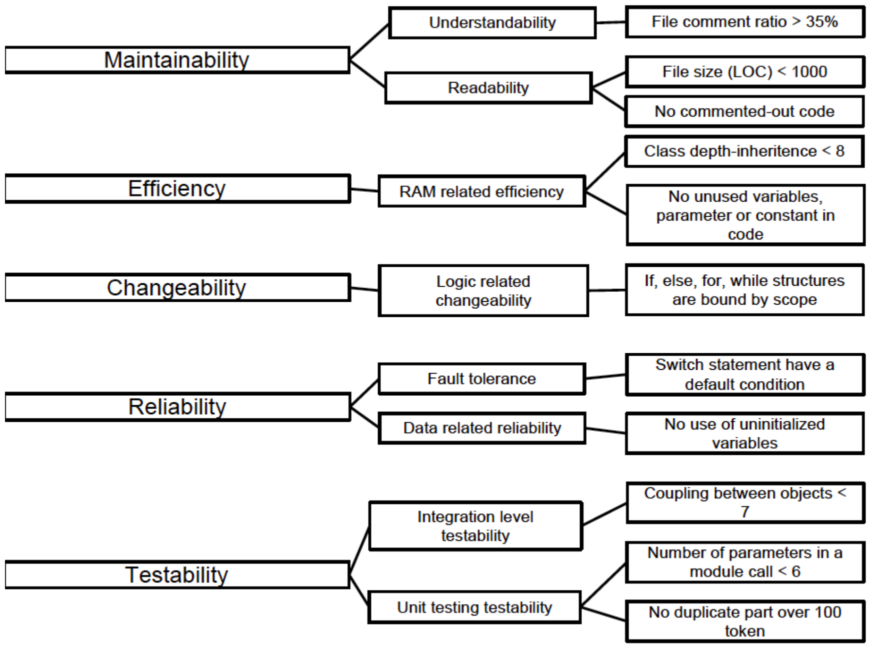

Figure 3.3. Hierarchical representation of the model described in TABLE 3.1

conformant to the quality objective. For instance, looking at TABLE I and Fig. 3.3, the remediation cost for the requirement of "Coupling between objects (CBO) < 7" is the cost to decrease the coupling from its current value X to 7. The remediation cost might not be constant but expressed by a remediation function. For example, reducing the coupling from 13 to 7 (-6) might cost a more than twofold effort of reducing it from 10 to 7 (-3). The total refactoring cost for a sub-characteristic is the sum of the remediation cost of each requirement violation.

Indices and Indicators

All of the SQALE indices represent the costs related to a given characteristic, estimated by adding up all of the remediation costs of the requirement violations of the connected sub-characteristics. For instance, in the example of TABLE 3.1, the SQALE Testability Index STI is the sum of the remediation costs of all violated requirements related to Integration level testability and Unit testing testability, i.e. "No duplicate part over 100 token", "Number of parameters in a module call < 6", "Coupling between objects < 7". The sum of all indexes is the Software Quality Index (SQI). It is also possible to obtain consolidated indices in the following way: the consolidated index of a given characteristic is equal to the sum of all the indices of the previous characteristics. For instance, the SQALE Consolidated Reliability Index (SCRI) is equal to STI + SRI, i.e. the sum of Testability and Reliability indexes. Moreover, a density index has to be associated with each absolute index dividing it by a measure representing the size of the artifact (lines of code, complexity, etc.). Finally the SQALE Method defines several synthesised indicators to summarize the overall quality status of the application: since indicators are out of the scope of the current work, we point the reader to the SQALE document for more detailed information.

3.3.2 Tailoring SQALE quality model to include Energy efficiency

We propose to tailor the SQALE model to include Energy Efficiency. As stated in the introduction, including Energy Efficiency in a quality model is an important step towards a measurable, repeatable and objective way to evaluate and improve the Energy Efficiency of a given application. We suggest introducing Energy Efficiency as a sub-characteristic related to the main characteristic "Efficiency". It cannot be a characteristic itself, because it is not an activity in the typical software lifecycle, but it is a sub-characteristic of second type (i.e. "a recognised taxonomy in terms of good and bad practices relating to the software's architecture and coding"). The next step is to identify appropriate code requirements that can be applied to evaluate the Energy Efficiency of a software product. For this purpose, in this section we propose a list of guidelines, derived from the literature [39] [61] [53], provided as solutions to developers in order to produce energy-efficient software. From these guidelines, we extract a set of proper requirements to be included in the SQALE Quality Model. Most of the guidelines suggested in the literature are not strictly code-related, but rather recommending general programming techniques. As a consequence, we selected from the original list those requirements that can be traced to actual code structures. These are listed in TABLE 3.2.

Nr.	*Guideline*	*Explanation*
GD1	Decrease algorithm complexity	Despite the fact that different algorithms can complete the same task, the way the task is performed can be totally different. Reducing the algorithm complexity can lead to saving energy.
GD2	Use Event-Based programming	Event-based programming avoids a waste of resources involved in doing unnecessary operations. If polling cannot be avoided, it is advised to select a fair time interval.
GD3	Batch I/O	Buffering I/O operations increases Energy Efficiency; the OS can power down I/O devices when not used.
GD4	Reduce data redundancy	Storage and transportation of redundant data impacts Energy Efficiency
GD5	Reduce memory leaks	With memory leaks the application can stall or crash. This unpredictable behaviour can alter the energy consumption and, more generally, they must always be avoided.

Table 3.2: Guidelines that can be translated into SQALE requirements

Starting from the selected guidelines, we express the set of requirements for evaluating software Energy Efficiency. These requirements, as specified in the SQALE Model Definition Document, [62], must be:

- Atomic
- Unambiguous
- Non-redundant
- Justifiable
- Acceptable
- Implementable
- Not in contradiction with any other requirement
- Verifiable

Our approach, in line with the SQALE methodology, is based upon translating the guidelines into code patterns automatically detectable with static analysis tools. We propose an estimate, based upon the presence of particular implementations that may cause energy waste. Since requirements are meant by SQALE to be language-dependant, we use the Java language as a reference in this chapter. TABLE 3.3 contains the requirements identified and mapped to the guidelines they are derived from. This is not to be intended as an exhaustive list but a first step towards source code Energy Efficiency quantification.

Nr.	*Guideline*	*Nr.*	*Requirement*
GD1	Decrease algorithm complexity	R1	Halstead's Effort< K
GD2	Use Event-Based programming	R2	Nr. of polling cycles=0
GD3	Batch I/O	R3	Nr. of FileInputStream.read() method calls = 0 [8]
GD4	Reduce data redundancy	R4	Nr. of unused variables = 0
GD5	Reduce memory leaks	R5.1	Nr. of Dead Store issues per class = 0
			Nr. of String|Boolean|Integer|Double constructor = 0

Table 3.3: Requirements for Energy Efficiency

R1: Halstead's Effort $< K$

Halstead's Effort [43] is a technique for describing the structural properties of algorithms. This metric has been selected because it gives an estimation of algorithm complexity, which is language-dependant, but not implementation-dependant as other metrics commonly used in this field (such as McCabe's Complexity [68]). K is a parameter to be defined according to specific application domains and project characteristics.

R2: Nr. of polling cycles = 0

To date and to our knowledge, no static analysis tool is able to detect the polling cycle, because polling structures can be implemented in various ways. However, we decided to keep this requirement and to devote further work to find a relevant metric to detect polling.

R3: Nr. of FileInputStream.read() method calls = 0

This requirement derives from a particular issue regarding the FileInputStream.read() method, that triggers a direct call to the underlying OS. If inserted into a cycle, it will realize an inefficient I/O policy. The use of a BufferedReader greatly improves performance and supposedly Energy Efficiency of the operation [69]. For example, the code shown in Listing 3.1 continuously calls the read() method of a FileInputStream object, thus triggering a large number of RPC calls to the Operating System.

Listing 3.1. Example of inefficient I/O policy

```
FileInputStream fis = new FileInputStream(filename]);
int cnt = 0;
int b;
while ((b = fis.read()) != -1)
{
        if (b == '\n')
                cnt++;
}
fis.close();
```

The code shown in Listing 3.2 makes use of a BufferedInputStream, which reads larger chunks of data than the FileInputStream. This greatly reduces Remote Procedure Calls, which improves Energy Efficiency by allowing the Operative System (OS) to turn off the I/O device when not needed.

Listing 3.2. Example of efficient I/O policy

```
FileInputStream fis = new FileInputStream(filename);
BufferedInputStream bis = new BufferedInputStream(fis);
int cnt = 0;
int b;
while ((b = bis.read()) != -1)
{
    if (b == '\n')
            cnt++;
}
bis.close();
```

R4: Nr. of unused variables = 0

The code in Listing 3.3 shows an example of Unused Field issue: the AClass contains a private field named "b", which is never used (the class does not provide a get() method for that field).

Listing 3.3. Example of Unused Field

```
private class AClass
{
        int a;
        int b;

        public int getA(){return a;}
}
```

An optimisation of this code would be providing a get() method for the "b" field, or removing the field if unnecessary.

R5.1: Nr. of Dead Store issues = 0

The code shown in Listing 3.4 contains a Dead Store issue, which means assigning a value to a local variable which is not read by any subsequent instruction.

Listing 3.4. Example of Dead Local Store

```
public int DeadLocalStore(int x)
{
        int constant_a = x;
        constant_a = 3

        return constant_a + x;
}
```

In the code above, x is stored to *constant_a*, but it is overwritten in the subsequent code line. A more efficient code is shown in Listing 3.5.

Listing 3.5. Example of refactored Dead Local Store

```
public int noDeadLocalStore(int x)
{
        int constant_a = 3;
        return constant_a + x;
}
```

The value of x is no longer stored to *constant_a* and then replaced.

The requirements specified above are derived from the guidelines [53] [61] of good programming practices provided in the literature. However, it is worth mentioning that such guidelines, despite being intuitive and acknowledged as effective by software industry specialists [103], did not receive any empirical validation. For this reason, and in order to make the choice of the above specified requirements *justifiable*, an empirical validation that quantitatively assesses their impact on Energy Efficiency is needed.

The last steps in the introduction of Energy Efficiency into SQALE are: tailoring the analysis model, tailoring the indices and the indicators. These steps are more straightforward then the previous one. Regarding the analysis model, a remediation function for each requirement violation ought to be defined in order to obtain the remediation cost. An example of remediation function for the requirement

$$\text{number of dead stores} > 0$$

could be:

$$10 + 2x$$

where x is the number of deadstores, 10 is the cost (in time units) of running a static analysis tool to detect them and 2 is the estimated time cost to review and refactor each dead store. We plan to estimate the remediation cost of each requirement violation through controlled experiments (e.g. observing the time required by subjects for a refactoring action) and questionnaires (i.e. asking directly to practitioners for estimations of refactoring actions). Energy Efficiency being a sub-characteristic of efficiency, the sum of the remediation costs of all its source requirements will be added to the total cost of the other efficiency sub-characteristics, obtaining the SQALE Efficiency Index. Finally, the indicators do not need tailoring because they are at the highest level of the quality model.

3.3.3 Considerations and Future Work

Energy efficiency is becoming a key factor in software development, given the ubiquity of software in everyday life and its hardware-related power consumption. Moreover, in devices running on batteries, efficient energy consumption is a key aspect. For this reason we propose introducing Energy Efficiency into the existing quality models. We selected SQALE, whose quality model is derived from the ISO/IEC 9126 and is strictly related to the software lifecycle activities. We tailor SQALE inserting Energy Efficiency as a sub-characteristic of efficiency, and we propose a set of specific requirements for the Java language starting from guidelines currently developed in the literature. The requirements identified are:

- Halstead's Effort $< K$
- Nr. of polling cycles $= 0$
- Nr. of FileInputStream.read() method calls $= 0$
- Nr. of dead store issues per class $= 0$
- Nr. of unread variables $= 0$

- Nr. of String|Boolean|Integer|Double constructor = 0

We identified two major challenges in requirements elicitation:

1. The translation of the guidelines in measurable requirements, whose violations are automatically identifiable by tools;

2. The validation of the requirements violation on energy consumption.

Future work will be devoted to execute the experiment to empirically validate the requirements, estimating both their negative impact on power consumption and the related remediation costs. We will also investigate whether other requirements are eligible to be included in the quality model under the Energy Efficiency sub-characteristic.

The next chapter will analyse the energy consumption profiling in data centres, desktop computers, and mobile handsets.

Chapter 4

Empirical Studies

This chapter contains the empirical studies carried out during this PhD programme, and it covers three Research Goals:

- RG3: In Sections 4.1, 4.2, and 4.3
- RG4: in Section 4.4
- RG5: In Section 4.5

Over the years, the use of Information Technology has exploded and IT has also contributed to environmental issues: the total electricity consumption by servers, computers, monitors, data communication equipment, etc. is increasing steadily [45]. According to [103], the ICT sector is responsible for a value between 2% and 10% of worldwide energy consumption. Therefore, it is necessary to improve awareness in the IT industry with regard to environmental problems, and this aspect should be considered from the academic point of view [102]: in fact, turning on research universities into living laboratories of the greener future [29], will permit us to quickly develop best practices and to make them available to industry and society in general. As an example we cite the work of Chiaraviglio et al. [24]: they applied a fully automatic measurement that is able to scale and track the number of devices powered on in real time. This technique has been applied at our same university, Politecnico di Torino. They created PoliSave, a software to turn a PC on/off by connecting directly to a website. PoliSave is being extended to all PCs in the Campus, with the goal of saving about 250,000 per year from the University energy bill. The study we present here is instead focused on the analysis of power consumption data, and it is designed to find out usage patterns of IT devices' energy consumption and to identify situations in which there is a waste of energy. Pinckard and Busch [85] also collected data on devices, focusing on the after-hours power state of networked devices in office buildings: they showed that most of devices are left powered on during the night, concluding that this is the first cause of energy waste. Usage analysis is a crucial step to optimize the energy consumption: this task is even more necessary within data centres where the number of computers is large. In this field, Bein et al. [14] tried to improve the energy efficiency of data centres: they studied the cost of storing vast amounts of data on the servers in a data center and they proposed a cost measure together with an algorithm that minimises such cost. In our analysis the number of PCs is lower, but we observe data from a real case [100].

4.1 Datacentres

4.1.1 Context of the analysis

One of the strategic goals of Politecnico di Torino is the green footprint cutting and related costs reduction. Managers know that whenever a change is needed, the first step is to figure out the current scenario in a quantitative way, that means to measure [27]. Starting from the indicators, it is then possible to find solutions, improve results and solve problems. Therefore, several measures should be present on the dashboard of the green power manager, one of them is the electrical power consumption of devices: for this reason, our University decided to install in several departments sensors to monitor the power consumption of rooms, lighting and conditioning systems, data elaboration centers and single IT devices such as servers, printers, switches. We were involved in the measurement process of such data in a research center affiliate to Politecnico di Torino, the Istituto Superiore Mario Boella (ISMB), and we present in this paper data collected and some facts found.

Instrumentation

The measurement of power consumption was done through a power monitoring system provided by an industrial partner. The system is composed by sensors inserted between the monitored devices and the electrical plugs to which they are plugged in. For entire sections of lighting and conditioning systems, instead, the sensor is applied directly on the conductor through a pincer. Both type of sensors can compute active and reactive power, voltage, current intensity, cos , with a desired sampling time (we selected ten minutes). Data collected by sensors are sent to a bridge through ZigBee, then the bridge forwards the data via Ethernet/Internet to the central servers. Data are then accessible on a web portal and can be exported to be analyzed. We monitor the active power consumption on the following devices of the ISMB research center:

- Three distinct servers:

- Server 1 (from 22nd April 2010 to 23rd November 2010)
- Server 2 (from 22nd April 2010 to 23rd November 2010)
- Server 3 (from 7th May 2010 to 23rd November 2010)

- A printer (from 5th March 2010 to 23rd November 2010)
- The conditioning system CED1, that is cooling the room where Server 1 and 2 are located (from 23rd November 2009 to 23rd November 2010)
- The lighting system in Server 3 Room (from 24th November 2009 to 23rd November 2010)

We list in Table 4.1 the characteristics of the three servers. Server 1 and Server 2 are both used as web servers: they host websites of research projects, where researchers share documents and files. Server 3 instead is used both as web server and to perform graphical operations. The printer is HP Laserjet P3005dn, with an operational power supply of 600W and standby consumption of 9W. Unfortunately we do not have information on the conditioning and lighting system. Finally, we define “instant power consumption as the average power consumption consumed in the sampling unit time (ten minutes).

	Server 1	**Server 2**	**Server 3**
Type	Dell PE r300	Dell PE1950 III	Dell Precision T5400
RAM	8 GB DDR 2	4 GB	4 GB
Proc	Quad Core Xeon X5460 3.16 GHz 64 bits	Quad Core Xeon E5410 2.33GHZ 32 bits	Dual Core Xeon 5200 2.49 GHz 64 bits
Power supply	400 W	670 W	875 W
Operating system	Windows Server 2003 R2 Enterprise X64	Ubuntu 2.6.24-19-server	i)Windows Server 2008 ii)Ubuntu 10.04 Server iii)Windows XP
Energy certification	NO	NO	Energy Star 4.0

Table 4.1: Energy Metrics and Benchmarks

Research questions

Eight different research questions drive data analysis. Firstly we have a group of questions (Overview) that is very general, and aims at discovering what is the actual average instant power consumption of the equipment in the ISMB research centre, that we suppose to be the typical equipment of similar centres. Overview questions are listed below:

1. What is the average instant power consumption of the servers in the last year?

2. What is the average instant power consumption of the printer in the last year?

3. What is the average instant power consumption of the other equipment (light, conditioning)?

   After that, we focus our analysis on the power consumption of servers, because we can reduce their consumption only by understanding how and how much they consume. The first question that is raised is whether the power consumption of the three servers is the same or not:

4. Are there differences between the servers instant power consumptions?

   Assuming, from the exploratory data analysis, that the power consumption in the studied context is not following any well-known distribution, we answer question 5 performing the Wilcoxon Two Sample test [9]. The difference we try to find with this question is an inter-server difference: the next step is to explore the aspects related to the progress of the single servers' power consumptions. Initially, we investigate whether the power consumption is homogeneous or variant:

5. Are there any peaks in instant power consumption or is it homogeneous? If so, how long do they last? Are they relevant, in terms of power values?

   We answer this question in a qualitative way, i.e. plotting, for each server, 2 different graphs: the power consumption over time and the estimated probability density. The first plot lets us identify the presence of peaks and trends in the observed time window, whilst the latter permits us to see if power consumption accumulated by peaks is relevant, looking to the frequency of the values associated to the peaks. Peaks represent a rapid growth or decrease, or deviations from a normal behaviour. However, a server could have several behaviours in terms of power consumption, associated, for example, to a different load or a particular software or hardware configuration. Therefore the scope of the next question is to understand the existence of different power consumption "behaviours", that we call "profiles".

6. Can we identify different usage profiles (e.g. active/standby)?

   We perform a cluster analysis to answer question 7. We use the K-Means algorithm [96] and the bivariate plots, obtained through normalisation and rescaling of the variables (watt-time couples). The selected clustering algorithm aims to group and find aggregations of data around certain values called "centres, which could point out different power consumption "profiles" and relationships between the variables (for instance, a typical profile in common computers is the standby profile).

   The plots let us visualize and verify the clusters found. If profiles are found, it is also important to verify if they correspond to daily/nightly activities, relating, for each server, the progress of power consumption with timetables of human activities in ISMB. Hence, we plot, for each server, the power consumption in a whole day, selecting for each week of the last 3 months a random day between Tuesday, Wednesday and Thursday (we avoid weekends, Mondays and Fridays because typically in these days human activities are not representative of the typical work day). Observing the 12 plots obtained and interviewing people working in the centre, we are able to identify the time range in which the majority of activities on all the 3 servers are carried out, that is between 9 a.m. and 8 p.m. At this point, it is possible to divide data into daily and nightly consumption, and then compare the two subsets, using the Wilcoxon Two Sample test, since data is not normally distributed. Moreover, we are interested in pointing out the power consumption of each profile, in order to understand how much energy is saved/lost by applying the configurations and conditions that determine the different power profiles. This is done by tagging each observation with the profile it belongs to and then summing up the cumulated consumption. Therefore, the research question is:

7. How much total energy did servers consume in the last year in the different profiles?

   Finally, the same question is replicated to the printer, which has two well-known profiles: an active profile when it is printing, and a stand-by profile when it is not.

8. How much energy can we save by turning off the printer when it doesn't work?

   All questions are about power, and related data are expressed in Watt (W), with an exception given for RQ 7, and RQ 8 that measure energy (KWh).

Threats to validity

The first threat of this research is an external threat: the analysis is performed on specific machines, thus generalising these results is not possible. However, it can be possible to look at this equipment as representative of a category of equipment with similar characteristics. Further, a derived internal threat is that the information on the technical characteristics of the IT equipment (printer, servers) and on their usage (massive, constant, etc) could be not enough to deeply motivate all the curves of the power consumption analysed and determine with precision the impact on the measures. Therefore, the causes that we derive from the observation, can be biased. Finally, we also identify a conclusion threat determined by the sampling time (ten minutes): as a consequence we have average values even for instant power consumption measures, and we could miss some fluctuations.

4.1.2 Analysis Results

- **Overview(RQ1 to RQ3)** We provide on Table 4.2 two descriptive statistics about RQ1 to RQ4: the average instant power consumption and the index of dispersion that quantifies how much data is sparse around the mean.

- Servers (RQ4 to RQ7)

  - **RQ4: Are there differences between the servers instant power consumptions**
    We observe in Table 4.3 that power consumptions of Server 1 is very different from the consumptions of Server 2 and 3. However, Server 2 and 3, even if similar in mean values (difference is only 8 Watts), have statistically different mean power consumptions.

  - **RQ5: Are there any peaks in instant power consumption or is it homogeneous? If so, how long do they last? Are they relevant, in terms of power values?**
    Looking to the time plots, we observe for Server 1 (Figure 4.1) many high spikes (that reach values that are more than the 50% of the mean), two low spikes and frequent switches between low and high values. However, the index of dispersion (see Table 4.2) is reduced, which means that the time duration of the peaks is short. For Server 2 (Figure 4.2), data is more concentrated around the mean value, and the peaks (3 low spikes and a dozen of high peaks) lasts for short periods of time (but longer than Server 1, as the index of dispersion suggests).

RQ	Device	Average Instant Power Consumption	Index of dispersion (var/mean)
1	Server 1	108.02 W	0.55
1	Server 2	145.12 W	2.3
1	Server 3	139.63 W	22.44
2	Printer	13.46 W	199.95
3	Light(Server 3)	107.76 W	877.56
4	Conditioning (Server 1+2)	2713.77 W	880.12

Table 4.2: Average instant power consumption of selected devices

Comparison	95% Difference Confidence Interval	P- val	Different?
Server 1 vs. Server 2	{ -37.78 , -37.68 }	< 2.2e-16	YES
Server 1 vs. Server 3	{ -47.57 , -47.36 }	< 2.2e-16	YES
Server 2 vs. Server 3	{ -8.43 , -8.33 }	< 2.2e-16	YES

Table 4.3: Result of Wilcoxon test on servers instant power consumptions

Finally, for Server 3 (Figure 4.3) the situation is yet different: it has a higher consumption and many high peaks until the end of August, then lower power consumption and peaks starting from September 2010. The change in the curve has a motivation: the server was used to perform continuous intensive tasks as image processing, parallel coding and massive video/audio streaming until end of August. Then, it was used as a normal web server, as Server 1 and Server 2. The variability of data has also the same behavior: higher until September, then reduced. Moreover, there is a very long period (about 20 days) of zero power consumption (it was powered down), followed by 4 other smaller periods of zero consumption. Plotting instead the probability density estimation of the servers' power consumptions, we can see around which values data is concentrated, and therefore if peaks are relevant both in terms of power consumption and duration. Server 1 (Figure 4.4) has 3 main concentrations of data: the highest is around the mean value (108 W), then there is a similar peak at

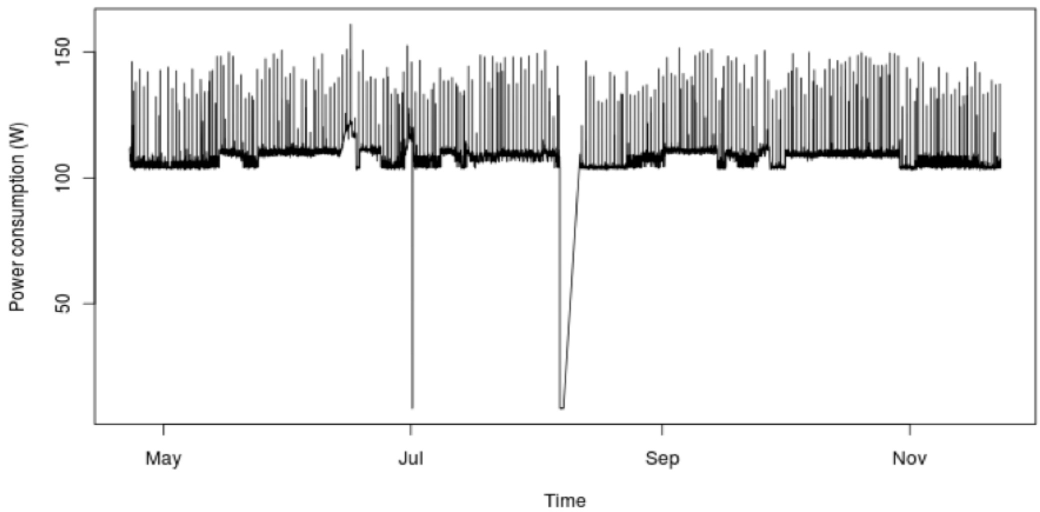

Figure 4.1. Instant power consumption over time (Server 1)

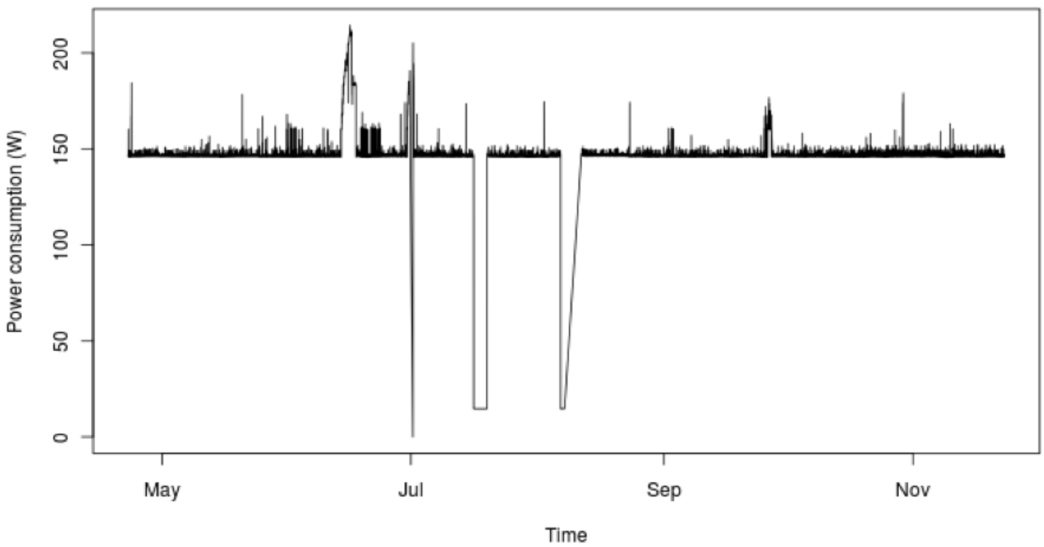

Figure 4.2. Instant power consumption over time (Server 2)

about 112 W and a lower one in their middle, finally two very low peaks at the two extremes of the graph. We conclude that peaks of Server 1 are relevant in terms of duration, but not in terms of variation from the mean value. The probability function of Server 2 (Figure 4.5) is totally different: data is concentrated around the mean value, and the distribution is very similar to a normal distribution with very low variance.

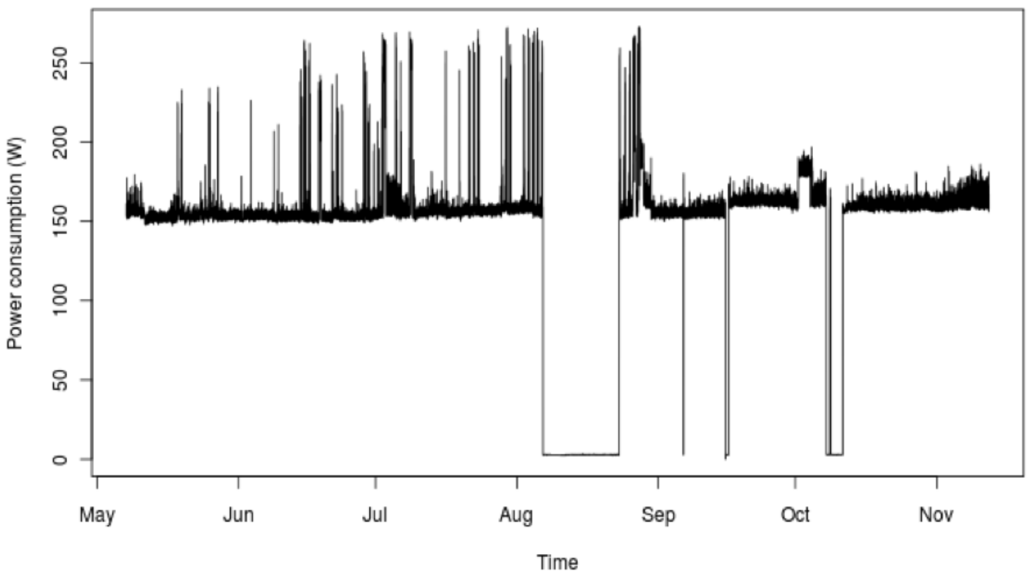

Figure 4.3. Instant power consumption over time (Server 3)

The higher index of dispersion is due to the small peak toward the 10W and the other one on the right of the mean. We observe that peaks for Server 2 are irrelevant in terms of duration, but some are quite far from the mean. Finally, we observe Server 3 in Figure 4.6: except the peak around the mean, the 2 big peaks (0 W and 160 W) have high probabilities, whilst the small peak on the right is quite far from the mean. As a consequence, spikes of Server 3 are relevant both for time length and power consumption. This concludes the answer of RQ6.

– **RQ7: Can we identify different usage profiles (e.g. active/standby)?**
We obtain from the K-Means algorithm 5 clusters for Server 1, 4 clusters for Server 2 and 4 Clusters for Server 3. The centers of the clusters are the following, in increasing order of power (W) :
  * Server 1: 8.36, 105.23, 106.97, 109.44, 110.47
  * Server 2: 14.64, 146.44, 146.68, 149.22
  * Server 3: 2.80, 154.51, 160.56, 246.81

We can surely identify a "low power profile" for all the servers (the lowest-value centre). Instead, active profiles could be more than one, especially for Server 1 (data varies from 105 to 110 W) and Server 3 (where the difference is clear, since values go from 154 to 246 W). For this reason, we perform a further cluster analysis, focused only on the active profile,

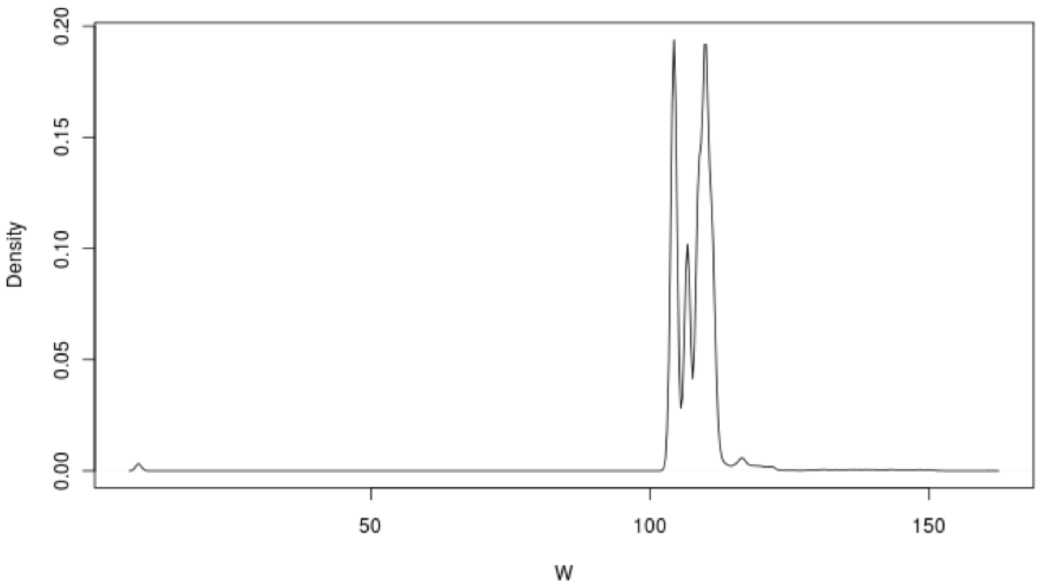

Figure 4.4. Probability density function estimation (Server 1)

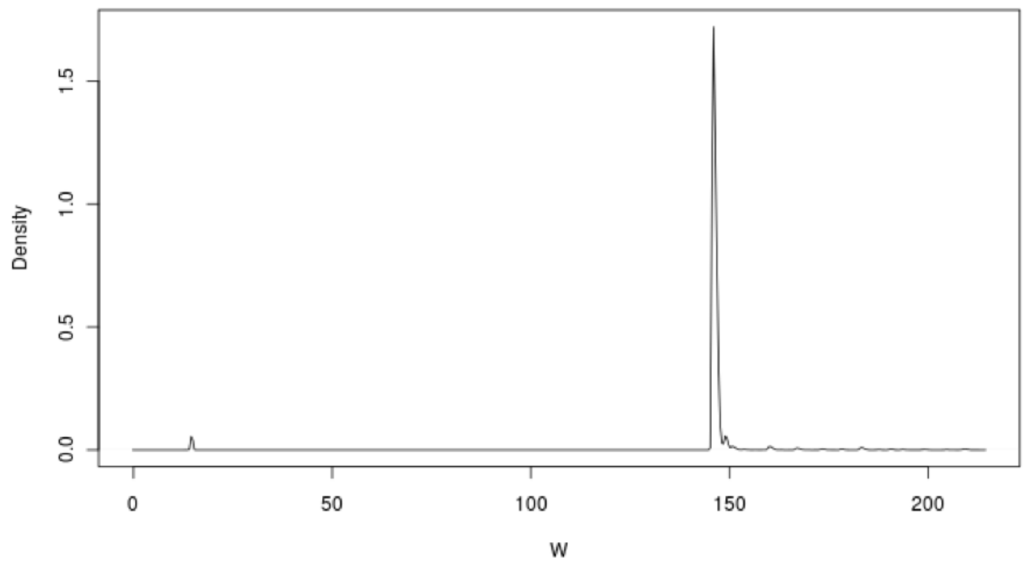

Figure 4.5. Probability density function estimation (Server 2)

which allows us to gain more information. We find the following centres:

* Server 1: 104.93, 105.21, 105.92, 109.76, 111.10, 138.61
* Server 2: 146.46 , 146.63, 185.10

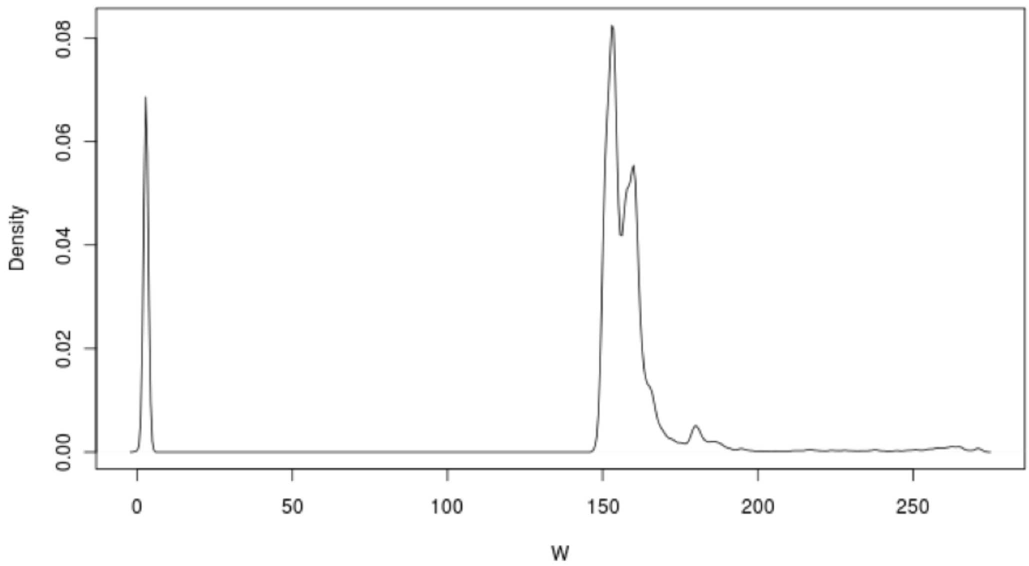

Figure 4.6. Probability density function estimation (Server 3)

* Server 3 : 154.27, 160.64, 245.87

Reducing to significant values, we can identify for all the servers at least two different high power profiles and a low power profile: we show values in Table 4.4. Subsequently, we investigate whether the two high power profiles are related to the day/night human activities. Even if the Wilcoxon statistical test (Table 4.5) verifies the difference between daily and nightly power consumption with the standard level of confidence of 95%, this difference is in practice negligible, since it is in the order of W for Server 1 and Server 2 and it is less then 1 W for Server 3.

- **RQ8: How much total energy did servers consume in the last year in the different profiles?**
  The estimated cumulated energy consumption of servers in the different power profiles is shown in column kWh of Table 4.6, whilst the column % shows the percentage of each cumulated power with respect to the total.

- Printer (RQ8)

  - RQ9: How much energy can we save by turning off the printer when it doesn't work?

    This is the overall data of the energy consumption of the printer in the Active Standby profiles:

* Active: 11.93 kWh
* Standby: 21.29 kWh

The difference is important indeed, 2/3 of energy are used (wasted) in the standby mode.

Servers	**High power profile 1 (W)**	**High Power Profile 2 (W)**	**Low Power Profile (W)**
Server 1	∼138	∼105	∼8
Server 2	∼185	∼146	∼14
Server 3	∼245	∼160	∼3

Table 4.4: Servers' power consumption profiles - data clustering

Servers	**95% Diff C.I. Between Day/Night (W)**	**P- val**	**Different?**
Server 1	{-1.00 e-05, -1.80 e-05}	0.05	YES
Server 2	{3.86 e-05 , 4.22 e-05}	0.01	YES
Server 3	{ 0.625 , 0.937 }	< 2.2e-16	YES

Table 4.5: Servers' power consumption comparison between day and night

Servers	**High power profile 1**		**High power profile 2**		**Low power profile**	
	kWh	**%**	**kWh**	**%**	**kWh**	**%**
Server 1	20.37	3.75	523.05	96.21	0.22	0.04
Server 2	33.57	4.62	692.2	95.17	1.57	0.22
Server 3	89.09	13.82	554.25	85.98	1.29	0.2

Table 4.6: Cumulative power consumption by profiles

4.1.3 Considerations

We analysed the power consumption in the last months of the equipment in the research center ISMB, which is affiliated to our University. We monitored at a high level data about conditioning and lighting systems and general devices and we conducted a more detailed analysis on the servers and the printer. We draw from the statistical analysis and the answers of the research questions the following facts, and related further questions for future work.

Servers

Fact 1. We found differences between the power consumptions of the three servers (RQ4), likely determined by software usage, not by hardware equipment. In fact, despite Server 1 has a more powerful hardware equipment (CPU, memory), it has the lowest power consumption.

Fact 2. The power consumption of servers is not homogeneous over time (RQ5). There are several peaks. Peaks are determined by software usage: as a matter of fact, Server 3 consumes up to 75% more when it is used for graphical operations.

Fact 3. Servers have different power consumption profiles (RQ7). This is determined by software usage.

Fact 4. Conditioning and lighting for servers consume more than computation (especially conditioning, that consumes approximately ten times more) (RQ3).

Fact 5. Low power profile (or Stand-by) for servers is useless ($< 1\%$) (RQ7).

Fact 6. There is no substantial difference between day and night servers' power consumption (RQ7).

Printer

Fact 7. The printer consumed more energy in standby mode than in active mode (RQ9).

Indeed, in our analysis, with a mechanism able to turn off the printer when it doesn't work, 21 kWh would have been saved in the period March-November 2010, which is 64% of the printer's total power consumption in that time range. Or, alternatively, shutting down the printer during the night, it is possible to save the standby power consumption (13 W, despite the 9W declared in the technical sheet), which means saving 47.45 KWh per year.

Even if this analysis is very initial, and specific to a few machines that may not be representative of the whole population, we believe it points out some simple checks that every energy manager should do as a first step to reduce energy consumption: consumption of conditioners and lighting, consumptions of printers in idle mode, consumption of servers both over day and night. Moreover, data we have presented could be compared to other analysis on equipment with similar characteristics. Future work will be devoted to understand more the reasons of the behaviours observed; notably to investigate deeper the motivations of the differences of servers' power consumptions, and to verify them experimentally by setting up different configurations/conditions in the machines to evaluate their impact on power consumption. Secondly, we will repeat the same analysis on the Data Elaboration Centre of our university, and we will compare it with the data presented in this paper. We think that the research questions that drove this analysis could be adopted and answered by other researchers in different universities and centres: building up a common benchmark of power consumption, it is possible to identify common and efficient solutions that can then be exported in industry and society.

4.2 Desktop PC

4.2.1 Background

In 2011, a post appeared on the MSDN Blog[1] : it concerned the energy consumption measurement of internet browsers. Authors measured power consumption and battery life of a common laptop across six scenarios and different browsers. They allowed each scenario to run for 7 minutes and calculated the average power consumption over that duration. The different scenarios were: Browsers navigated to about:blank (power consumption of the browser UI), loading a popular news Web sites (common HTML4 scenario), running the HTML5 Galactic experience (representative of graphical HTML5 scenario) and fish swimming around the FishIE Tank (what test is complete without FishIE?). The baseline for scenarios comparison was the Windows 7 without any browsers running. Authors ran *IE9*, *Firefox*, *Opera* and *Safari* for each scenario and then they made a comparison of the obtained results. They executed the same operations with the different browsers, obtaining very different results on power consumption and laptop battery life [82] [81].

[54] presented a solution for VM power metering. Since measuring the power consumption of a Virtual Machine is very hard and not always possible, authors built power models to get power consumption at runtime. This approach was designed to operate with low runtime overhead. It also adapts to changes in workload characteristics and hardware configuration. Results showed 8% to 12% of additional savings in virtualized data centres. Another related work is PowerScope [31]: this tool uses statistical sampling to profile the energy usage of a computer system. Profiles are created both during the data collection stage and during the analysis

[1]Browser Power Consumption - Leading the Industry with IE 9, `http://blogs.msdn.com/b/ie/archive/2011/03/28/browser-power-consumption-leading-the-industry-with-int`

stage. During the first stage, the tool samples both the power consumption and the system activity of the profiling computer and then generates an energy profile from this data without profiling overhead. During data collection, authors use a digital multimeter to sample the current drawn by the profiling computer through its external power input. After that, they modified Odyssey platform for mobile computing. When there is a mismatch between predicted demand and available energy, Odyssey notifies applications to adapt. This is one of the first examples of *Energy-Aware* software.
In 1995, the first attempts were made to profile the energy performance of a computer. Lorch [64] in his M.S. thesis explained that there are two aspects to consider while measuring the breakdown of power consumption on a portable computer: I) Measuring how much power is consumed by each component, II) Profiling how often each component is in each state.

Other works about profiling and measuring energy consumption are related to embedded systems. For instance, JouleTrack [93] runs each instruction or short sequences of instruction in a loop and measure the current/power consumption. The user can upload his C source code to a Web Server which compiles, links and executes it on an ARM simulator. Program outputs, assembly listing and the run-time statistics (like execution time, cycle counts etc.) are then available and passed as parameters to an engine which estimates the energy consumed and produces graphs of different energy variables. Results showed that the error of predictions was between 2% and 6%. The concept of energy-awareness is based upon a complete knowledge on how and where energy is consumed on a device. In [20], authors present a detailed analysis of power consumption in a mobile device, focusing on the hardware subsystems, through common and realistic usage scenarios. Results show that the GSM module and the display are the most power-consuming components: for example, a GSM phone call on OpenMoko Neo Freerunner, HTC Dream G1 and Google Nexus One consumes 1135 mW, 822 mW and 846 mW respectively.

Usually, an accurate power consumption analysis of mobile or embedded devices is component-based. However, instantaneous information about discharge current and remaining battery capacity is not always available, because most devices do not have built-in sensors to collect these data. In [108], a technique called PowerBooter is proposed to build a battery-based model automatically. Authors motivate this decision by considering that different mobile devices of the same category show different power consumption, and a specific power consumption model for each device is difficult to obtain. Thus, instead of using external metering instrumentation to detect power consumption, only the internal battery voltage sensor is used, which is found across many modern smartphones. Authors indicate that for a 10-second interval, the PowerBooter technique has an accuracy of about 4.1% within measured values.

From a software engineering point of view, most contributions are devoted to developing frameworks and tools for energy metering and profiling. Also in [108], authors propose an on-line power estimation tool called PowerTutor. It implements the PowerBooter model in order to profile power consumption of applications, based upon their component usage. Another example, which makes use of external metering devices, is ANEPROF [25], which authors define as a real-measurement-based energy profiler able to reach function-level granularity. It is developed for Android OS-based devices, thus it is aimed at profiling Java applications. It is based on JVM event profiling, using software probes to record runtime events and system calls. Authors had to address several design issues, such as overhead control and proper time synchronization. Power consumption profiling is made through correlation of real-time power measurements done by an external DAQ, connected to an ARM Computer-on-Module running Android 2.0. Authors also provide profiling data of four popular applications (Android Browser, GMail, Facebook, Youtube). The accuracy of ANEPROF depends on the hardware meter used. Its CPU overhead is stated to be less than 5%. Finally, SEMO [28] is a smart energy monitoring system, developed for Android, which also provides also application-level consumption monitoring. This system is composed of three components: an *inspector*, which monitors the information on the battery, warning users when the battery reaches a critical condition; a *recorder*, which basically logs the actual charge of the battery and the running applications, and an *analyzer*, which calculates the energy consumption rate for each application and ranks them according to it.

Another alternative for energy measurement is low-level power-analysis using instruction-level models [98]. These models provide accurate power estimates for small kernels of code. An example of this kind of model is presented in Equation 4.1 where [66] Energy is the total energy dissipation of the program.

$$Energy = \sum_i (BC_i) + \sum_{i,j} (SC_{i,j}\, N_{i,j}\,) + \sum_k (OC_k) \tag{4.1}$$

The first part is the summation of the base energy cost of each instruction (BC_i is the base energy cost and N_i is the number of times instruction i is executed). The second part accounts for the circuit state ($SC_{i,j}$ is the energy cost when instruction i is followed by during the program execution). The third part accounts for energy contribution OC_k of other instruction effects such as stalls and cache misses during the program execution.

The study presented here is instead focused on the analysis of power consumption data, and it is designed to find out usage patterns of IT devices' energy consumption and to identify situations in which there is a waste of energy. Pinckard and Busch [104] also collected data on devices, focusing on the after-hours power state of networked devices in office buildings: they showed that most of devices are left powered on during the night, concluding that this is the first cause of energy waste.

Usage analysis is a crucial step to optimise the energy consumption: this task is even more necessary within data centres where the number of computers is large. In this field, Bein et al. [14] tried to improve the energy efficiency of data centres: they studied the cost of storing vast amounts of data on the servers in a data centre and they proposed a cost measure together with an algorithm that minimises such cost.

4.2.2 Study Design

Goal Description and Research Questions

The aim of this research is to assess the impact of software and its usage on power consumption in computer systems. The goal is defined through the Goal-Question-Metric (GQM) approach. [99]. This approach, applied to the experiment, led to the definition of the model presented in Table 4.17. The first research question investigates whether and how much software impacts power consumption. Different applications and usage patterns will be tested. The second research question investigates whether a categorization of usage scenarios with respect to functionality, is also valid for power consumption figures. The third research question tries to find a quantifiable relationship between power consumption and actual usage of the computer system, by selecting four metrics relative to the main system resources (CPU, Disk, Memory and Network).

Goal	*Evaluate* *for the purpose* *with respect to* *from the viewpoint* *in the context of*	software usage of assessing its energetic impact power consumption of the System User Desktop applications
Research Question 8	Does software impact power consumption?	
Metric	Consumed Power (Watts)	
Research Question 9	Is it possible to classify software usage scenarios basing upon power consumption?	
Metric	Consumed Power (Watts)	
Research Question 10	What is the relationship between usage and power consumption?	
Metric	CPU Usage (percentage)	
Metric	Memory Usage (reads/writes)	
Metric	Disk Usage (reads/writes)	
Metric	Network Usage (Packets/sec)	
Metric	Consumed Power (Watts)	

Table 4.7: The GQM Model

Variable Selection

In order to answer the Research Questions, it is necessary to specify the independent variables that will characterize the experiment. The following usage scenarios, described in detail, will provide the basis for the analysis.

0 - Idle. This scenario aims at evaluating power consumption during idle states of the system. In order to avoid variations during the runs, most of OS'automatic services were disabled (i.e. Automatic Updates, Screen Saver, Anti-virus and such).

1 - Web Navigation. This scenario depicts one of the most common activities for a basic user - Web Navigation. During the simulation, the system user starts a web browser, inputs the URL of a web page and follows a determined navigation path. Google Chrome has been chosen as the browser for this scenario because of its better performance on the test system, which allowed us to increase navigation time. The website chosen for this scenario is the homepage of the SoftEng research group `http://softeng.polito.it`, in order that the same contents and navigation path could be maintained during all the scenario runs.

2 - E-Mail. This scenario simulates sending and receiving E-Mails. For this scenario's purpose, a dedicated E-Mail account has been created in order to always send and receive the same message. In this scenario, the system user opens an E-Mail Client, writes a short message, sends it to himself, then starts checking for new messages by pushing on the send/receive button. Once the message has been received, the user reads it (the reading activity has been simulated with an idle period), then deletes the messages and starts over.

3 - Productivity Suite. This scenario evaluates power consumption during the usage of highly-interactive applications, such as office suites. For this scenario, Microsoft Word 2007 has been chosen, the most used Word Processor application. During the scenario execution, the system user launches the application and creates a new document, filling it with content and applying several text editing/formatting functions, such as enlarge/shrink Font dimension, Bold, Italics, Underlined, Character and background colors, Text alignment and interline, lists. Then the document is saved on the machine's hard drive. For each execution a new file is produced, thus the old file gets deleted at the end of the scenario.

4 - Data Transfer (Disk). This scenario evaluates power consumption during operations that involve the File System, and in particular the displacement of a file over different positions of the hard drive, which is a very common operation. For this scenario's purpose, a data file of a relevant size (almost 2 GB) has been prepared in order to match the file transfer time with the prefixed scenario duration (5 minutes). The scenario structure is as follows: the system user opens an Explorer window, selects the file and moves it to another location. It waits for file transfer to end, then closes Explorer and exits.

5 - Data Transfer (USB). As using portable data storage devices has become a very common practice, this scenario has been developed to evaluate power consumption during a file transfer from the system hard drive to an USB Memory Device. This scenario is very similar to the previous one, except for the file size (which is slightly lower, near 1.8 GB) and the file destination, which is the logical drive of the USB Device.

6 - Image Browsing/Presentation. This scenario evaluates power consumption during another common usage pattern, which is a full-screen slide-show of medium-size images, which can simulate a presentation as well as browsing through a series of images. In this scenario, the system user opens a PDF File composed of several images, using the Acrobat Reader application. It sets the Full-Screen visualisation, then manually switches through the images every 5 seconds (thus simulating a presentation for an audience).

7 - Skype Call (Video Disabled). For an average user, the Internet is without any doubt the most common resource accessed via a Computer System. Moreover, as broadband technologies become increasingly available,it has been thought to be reductive not to consider usage scenarios that make a more intensive use of the Internet than Web Navigation and E-Mails. Thus, the Skype scenario has been developed. Skype is the most used application for Video Calls and Video Conferences among private users. For this scenario's purposes, a Test Skype Account was created, and the Skype Application was deployed on the test machine. Then, for each run, a test call is made to another machine (which is a laptop situated in the same laboratory) for 5 minutes, which is the prefixed duration of all scenarios.

8 - Skype Call (Video Enabled). This scenario is similar to scenario 7, but the Video Camera is enabled during the call. This allows an evaluation of the impact of the Video Data Stream both on power consumption and on system resources.

9 - Multimedia Playback (Audio). This scenario aims to evaluate power consumption during the reproduction of an Audio content. For this scenario's purpose, an MP3 file has been selected, with a length of 5 minutes, to reproduce through a common multimedia player. Windows Media Player has been chosen, as it is the default player in Microsoft systems, and thus one of the most diffused.

10 - Multimedia Playback (Video). Same as above, but in this case the subject for reproduction is a Video File in AVI format, same duration.

11 - Peer-to-Peer. As for the Skype scenarios, the decision has been made to also take into account a Peer-to-Peer scenario, which has proven to be a very common practice among private users. For this scenario, BitTorrent was selected as a Peer-to-Peer application, because of its large diffusion and less-variant usage pattern if compared to other Peer-to-Peer networks with more complex architectures. During this scenario, the system user starts the BitTorrent client, opens a previously provided .torrent archive, related to an Ubuntu distribution, and starts the download, which proceeds for 5 minutes. After every execution, the partially downloaded file is deleted, in order to repeat the scenario with the same initial conditions.

In Table 4.8 all the scenarios are summarized with a brief description of each of them. The last column reports the category which the scenarios belong to, from a functional point of view, according to the following:

- *Idle* (Scenario 0): it is the basis of the analysis, evaluates power consumption during the periods of inactivity of the system.

- *Network* (Scenarios 1,2,7,8,11): it represents activities that involve network subsystems and Internet.

- *Productivity* (Scenario 3): it is related to activities of personal productivity.

- *File System* (Scenarios 4,5): it concerns activities that involve storage devices and File System operations.

- *Multimedia* (Scenarios 6,9,10): it represents activities that involve audio/video peripherals and multimedia contents.

Nr.	Title	Description	Category
0	Idle	No user input, no applications running, most of OS'automated services disabled.	Idle
1	Web Navigation	Open browser, visit a web-page, operate, close browser.	Network
2	E-Mail	Open e-mail client, check e-mails, read new messages, write a short message, send, close client.	Network
3	Productivity Suite	Open word processor, write a small block of text, save, close.	Productivity
4	Data Transfer (disk)	Copy a large file from a disk position to another.	File System
5	Data Transfer (USB)	Copy a large file from an USB Device to disk.	File System
6	Presentation	Execute a full-screen slide-show of a series of medium-size images.	Multimedia
7	Skype Call (no video)	Open Skype client, execute a Skype conversation (video disabled), close Skype.	Network
8	Skype Call (video)	Open Skype client, execute a Skype conversation (video enabled), close Skype.	Network
9	Multimedia (Audio)	Open a common media player, play an Audio file, close player.	Multimedia
10	Multimedia (Video)	Open a common media player, play a Video file, close player.	Multimedia
11	Peer-to-Peer	Open a common peer-to-peer client, put a file into download queue, download for 5 minutes, close.	Network

Table 4.8: Software Usage Scenarios Overview

Moreover, as anticipated in the previous section, four metrics have been selected to evaluate the system usage. These metrics were measured by means of software logging (as will be explained in the *Instrumentation* section) considering the following values:

- CPU
  - CPU Time Percentage, intended as time spent by the CPU doing active work in a second
  - CPU User Time Percentage, intended as time spent by the CPU executing user instructions (i.e. applications) in a second

  - CPU Privileged Time Percentage, intended as time spent by the CPU executing system instructions (services, daemons) in a second
  - CPU Deferred Procedure Calls Percentage, intended as time spent by the CPU executing DPC in a second
  - CPU Interrupt Time Percentage, intended as time spent by the CPU serving interrupts in a second
  - CPU C1 Time Percentage, intended as time spent by the CPU in low-power (C1) State
  - CPU C2 Time Percentage, intended as time spent by the CPU in low-power (C2) State
  - CPU C3 Time Percentage, intended as time spent by the CPU in low-power (C3) State

- Memory
  - Memory Page Writings per second
  - Memory Page Readings per second
  - Memory Available (KiloBytes) per second
- Hard Disk
  - Physical Disk Transfers (Read/Write) per second
  - Logical Disk Transfers (Read/Write) per second
- Network
  - Network Packets per second as seen by the Network Interface Card

The dependent variable selected for the experiment is P i.e. the instant power consumption (W). Therefore, P_n is the average power consumption during Scenario $n = 1..11$ and $P_{idle|net|prod|file|MM}$ is the average power consumption of (respectively) Idle, Network, Productivity, File System and Multimedia scenarios.

Hypotheses Formulation

Based upon the GQM Model, the Research Questions can be formalised into Hypotheses. In order to formally express Research Question 10, $\rho(x, y)$ expresses the correlation coefficient between variables x and y. β represents a significant correlation value, which will be defined later in this Section.

- *RQ 8: Does Software impact Power Consumption?*

$H1_0$: $P_{idle} \geq P_n$, $n \in [1,11]$
$H1_a$: $P_{idle} < P_n$, $n \in [1,11]$

- *RQ 9: Is it possible to classify software usage scenarios based upon power consumption?*

$H2_0$: $P_{idle} = P_{net} = P_{prod} = P_{file} = P_{MM}$
$H2_a$: $P_{idle} \neq P_{net} \neq P_{prod} \neq P_{file} \neq P_{MM}$

- *RQ 10: What is the relationship between usage and power consumption?*

$H3_0$: $\rho(I_{CPU}, P) = \rho(I_{Memory}, P) = \rho(I_{Disk}, P) = \rho(I_{Network}, P) = 0$
$H3_a$: $max[\rho(I_{CPU}, P), \rho(I_{Memory}, P), \rho(I_{Disk}, P), \rho(I_{Network}, P)] > \beta$

Instrumentation

Every scenario has been executed automatically by means of a GUI Automation Software for 5 minutes, obtaining 30 runs per scenario, each composed of 300 observations (one per second) of the instant power consumption value (W).

The test machines selected are two Desktop PCs of different generations. In Table 4.9, the Hardware/Software configuration of the machines is presented. As can be seen, the difference in terms of hardware is relevant; this will allow us to make some evaluations about how power consumption varied over the years, with the evolution of hardware architectures.

	Desktop 1 (old generation)	**Desktop 2 (new generation)**
CPU	AMD Athlon XP 1500+	Intel Core i7-2600
Memory	768 MB DDR SDRAM	4 GB DDR3 SDRAM
Display Adapter	ATI Radeon 9200 PRO 128 MB	ATI Radeon HD 5400
HDD	Maxtor DiamondMax Plus 9 80GB Hard Drive	Western Digital 1 TB
Network Adapter	NIC TX PCI 10/100 3Com EtherLink XL	Intel 82579V Gigabit Ethernet
OS	Microsoft Windows XP Professional SP3	Windows 7 Professional SP1

Table 4.9: HW/SW Configuration of the test machine

Different software and hardware tools have been used to do monitoring, measurement and test automation. The Software tool adopted is Qaliber[2], (see Figure 4.7) which is mainly a GUI Testing Framework, composed of a Test Developer Component, that allows a developer to write a specific test case for an application, by means of "recording" GUI commands, and a Test Builder Component, which allows the creation of complex usage scenarios by combining the use cases. One of the most important features of Qaliber is its possibility to log system information during scenario execution, using Microsoft's Performance Monitor Utility. By defining a specific Counter Log, adding all the variables of interest, it is possible to tell Qaliber to start Performance Monitor simultaneously with the Scenario, thus allowing a complete monitoring of all the statistics needed for this analysis.

The measurement of power consumption was done through two different devices. For the old-generation PC, PloggMeter[3] (see Figure 4.8) device was used. This device is capable of computing Active and Reactive Power, Voltage, Current Intensity, $Cos\varphi$. The data is stored within the PloggMeter's 64kB memory and can be downloaded in a text file format via Zigbee wireless connection to a Windows-enabled PC or Laptop or viewed as instantaneous readings on the installed Plogg Manager

[2]Qaliber - GUI Testing Framework, `http://www.qaliber.net/`

[3]Youmeter - Plogg Technologies, `http://www.youmeter.it/youmeter`

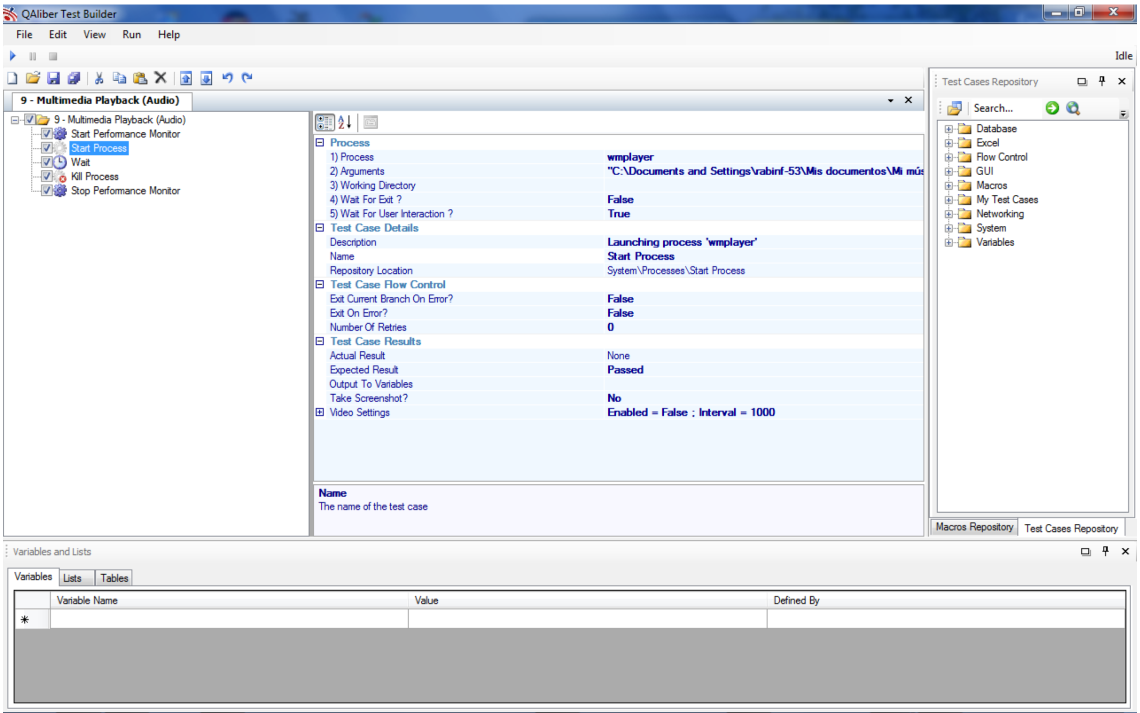

Figure 4.7. Qaliber Test Builder screenshot

software. The device drivers were slightly modified to adapt the PloggMeter recording capability to this analysis' purposes, specifically to decrease the logging interval from 1 minute (which is too wide if compared to software time) to 1 second.

Figure 4.8. The PloggMeter device

For the new-generation PC, WattsUp PRO ES[4] (see Figure 4.9) device was used. This device is capable of measuring current power consumption (Watts), power factor, line voltage and other metrics. The data is stored within the device's internal memory, and then downloadable via USB interface. The sampling rate resolution is 1 second.

Figure 4.9. The WattsUp Pro ES device

[4]WattsUp Pro ES, https://www.wattsupmeters.com/secure/products.php?pn=0&wai=0&spec=2

Analysis Methodology

The goal of data analysis is to apply appropriate statistical tests to reject the null hypothesis. The analysis will be conducted separately for each scenario in order to evaluate which one has an actual impact on power consumption.

In order to extract a Power Consumption profile for each Usage Scenario, a set of descriptive statistics was derived from the experimental data. For a single scenario, a total of 30 runs were executed, each composed of 300 observations (one per second) of the power consumption value. Thus, the calculations for the descriptive statistics were made using two approaches: firstly, the average of each run is extracted, obtaining a short vector of 30 elements, which was used as the subject of our analysis. This method allowed us to speed up the calculations, and because of the decreased sampling rate, the data was less variant and showed an almost regular distribution.

Afterwards, the same analysis on the full datasets was applied, which means a total of 9000 observations. Comparing the results from these two approaches, focusing on the Index of Dispersion and the variance, the variability of a single scenario can be appreciated, which was also a useful tool for validating the experiment.

First of all, the null hypothesis $H1_0$ will be tested for each scenario. Then the scenarios will be grouped into categories and $H2_0$ will be tested for each category.

First of all, data distribution must be analysed, in order to determine the appropriate testing method for each hypothesis. The data distribution analysis was conducted using the Shapiro-Wilk normality test. Since its results pointed out that the data was not normally distributed, non-parametric tests were adopted, in particular the Mann-Whitney test [46] for testing $H2_0$, and the Spearman's rank correlation coefficient (also known as Spearman's ρ) for testing $H3_0$.

The first hypothesis $H1_0$ is clearly directional, thus the one-tailed variant of the test will be applied. The second and third hypotheses $H2_0$, $H3_0$ are not directional, therefore the two-sided variant of the tests will be applied.

We will draw conclusions from our tests based on a significance level $\alpha = 0.05$, that is we accept a 5% risk of type I error – i.e. rejecting the null hypothesis when it is actually true. Moreover, since we perform multiple tests on the same data – precisely twice: first overall and then by category – we apply the Bonferroni correction to the significance level and we actually compare the test results versus a $\alpha_B = 0.05/2 = 0.025$. As regards Spearman's ρ significance, using 298 degrees of freedom (since 300 observations per scenario are available) the significance level of the ρ coefficient is 0.113. Thus, correlations' coefficients resulting higher than this value can be considered as significant positive or negative correlations.

Validity evaluation

The threats of experiment validity can be classified in two categories: **internal** threats, derived from treatments and instrumentation, and **external** threats regarding the generalization of the work.

There are three main internal threats that can affect this analysis. The first concerns the *measurement sampling*: measurements were taken with a sampling rate of 1 second. This interval is a compromise between the power metering device capability and the software logging service. However, it could be a wide interval if compared to software time.

Subsequently, *network confounding factors* could arise: as several usage scenarios involving network activity and the Internet are included in our treatments, the unpredictability of the network behaviour could affect some results. Another confounding factor is represented by *OS scheduling operations*: the scheduling of user activities and system calls is out of the experiment control. This may cause some additional variability in the scenarios, especially for those that involve the File System.

In addition, the two machines on which our tests are performed are different in terms of hardware and software configuration. This is done on purpose, because we wanted to test devices which could represent common machines used in home and office scenarios, for both generations. Thus, installing an old version of an operating system on a new machine or vice versa would have altered this assumption. However, this will introduce another confounding factor, but still, will provide useful information regarding the evolution of these systems, even if no specific research hypotheses can be verified about the comparison.

Finally, the main external threat concerns a possible *limited generalization* of the results: this is due to the fact that the experiment was conducted on only two different test machines, which is a limited population to be representative of a whole category.

4.2.3 Results

Preliminary Data Analysis

We present in Tables 4.10, 4.11 the following descriptive statistics about measurements for each scenario. Tables reports in this order mean (Watts), median (Watts), standard error on the mean, 95% confidence interval of the mean, variance, standard deviation (σ), variation coefficient (the standard deviation divided by the mean), index of dispersion (variance-to-mean ratio, VMR).

Power consumption shows an excursion of about 11 W for both PCs, even if the baseline is quite different (an average of 87 W in Idle scenario for the Old PC, 51 W for the New PC). Moreover, the very low variability indexes ensure that the different samples for each scenario are homogeneous.

	Old-Generation PC							
	Mean	**Median**	**S.E.**	**C.I.**	**Variance**	σ	**Var.Co.**	**VMR**
0 - Idle	86.81	86.69	0.007	0.013	0.424	0.650	0.007	0.005
1 - Web	89.09	88.57	0.011	0.022	3.372	1.836	0.021	0.038
2 - E-Mail	88.03	87.11	0.024	0.047	5.195	2.279	0.026	0.059
3 - Prod	90.12	89.40	0.025	0.500	5.862	2.421	0.027	0.065
4 - Disk	94.12	97.21	0.048	0.095	21.12	4.595	0.049	0.224
5 - USB	96.41	97.10	0.024	0.046	5.047	2.246	0.023	0.052
6 - Image	91.97	91.48	0.041	0.081	15.474	3.934	0.043	0.168
7 - Skype	91.87	91.69	0.015	0.029	1.981	1.407	0.015	0.022
8 - SkypeV	95.40	95.75	0.020	0.040	3.844	1.960	0.020	0.040
9 - Audio	88.14	87.94	0.013	0.025	1.429	1.195	0.013	0.016
10 - Video	88.61	88.57	0.009	0.017	0.677	0.823	0.009	0.008
11 - P2P	88.46	88.25	0.010	0.019	0.842	0.917	0.010	0.009

Table 4.10: Scenarios Statistics Overview: Old-Generation PC

	New-Generation PC							
	Mean	**Median**	**S.E.**	**C.I.**	**Variance**	σ	**Var.Co.**	**VMR**
0 - Idle	51.39	51.20	0.007	0.015	0.507	0.712	0.013	0.009
1 - Web	54.05	53.9	0.014	0.028	1.883	1.372	0.025	0.035
2 - E-Mail	53.40	53.40	0.011	0.021	1.123	1.059	0.019	0.021
3 - Prod	53.09	52.70	0.016	0.032	2.369	1.539	0.029	0.044
4 - Disk	60.24	62.10	0.037	0.072	12.38	3.518	0.058	0.205
5 - USB	61.29	61.90	0.023	0.046	4.901	2.214	0.036	0.080
6 - Image	52.75	52.50	0.011	0.023	1.214	1.102	0.021	0.023
7 - Skype	56.23	56.30	0.016	0.032	2.420	1.555	0.027	0.043
8 - SkypeV	62.13	62.90	0.036	0.070	11.428	3.380	0.054	0.184
9 - Audio	52.87	52.70	0.006	0.012	0.315	0.561	0.010	0.006
10 - Video	54.14	54.00	0.007	0.013	0.420	0.648	0.012	0.008
11 - P2P	54.32	54.50	0.008	0.016	0.609	0.780	0.014	0.011

Table 4.11: Scenarios Statistics Overview: New-Generation PC

Hypothesis Testing

The results of hypotheses testing of the research questions are exposed in this section.

The testing of hypothesis H_1 and H_2 are exposed in Table 4.12 and 4.13. These table report the scenarios tested, the p-value of Mann-Whitney test and the estimated difference of the medians between Idle scenario and the other ones.

In Figure 4.10 is shown the Bar Plot of the Power Consumption values, which provide a graphical and immediate comparison between the different scenarios' Power Consumption profiles. In Figure 4.11 is shown the Bar Plot of the Power Consumption increase (in watts), with respect to idle, of each scenario. Figure 4.12 shows the Box Plot of scenario categories for each PC. As regards hypothesis H_3, which evaluates correlations between resource usage and power consumption, more steps are needed. First of all, Table 4.14 reports the results of the Data Distribution Analysis. Then, in Tables 4.15 and 4.16, are presented the results of the correlation test using Spearman's method, with a 95% confidence interval, applied to every couple $(watt, variable)$ for each scenario. Only the significant coefficients are listed.

Question 1: Does software impact power consumption?

$H1 : P_{idle} \neq P_n \forall n \in [1,11]$.

	Old-Gen PC		*New-Gen PC*	
Scenario Comparison	**p-value**	**Est. Diff**	**p-value**	**Est. Diff**
0 - Idle vs. 1 - Web Navigation	< 0.0001	-1.87	< 0.0001	-2.60
0 - Idle vs. 2 - E-Mail	< 0.0001	-0.52	< 0.0001	-2.10
0 - Idle vs. 3 - Productivity Suite	< 0.0001	-2.71	< 0.0001	-1.50
0 - Idle vs. 4 - IO Operation (Disk)	< 0.0001	-10.41	< 0.0001	-10.80
0 - Idle vs. 5 - IO Operation (USB)	< 0.0001	-10.41	< 0.0001	-10.60
0 - Idle vs. 6 - Image Browsing	< 0.0001	-4.69	< 0.0001	-1.20
0 - Idle vs. 7 - Skype Call (No Video)	< 0.0001	-5.10	< 0.0001	-5.00
0 - Idle vs. 8 - Skype Call (Video)	< 0.0001	-9.05	< 0.0001	-11.50
0 - Idle vs. 9 - Multimedia Playback (Audio)	< 0.0001	-1.25	< 0.0001	-1.50
0 - Idle vs. 10 - Multimedia Playback (Video)	< 0.0001	-1.87	< 0.0001	-2.80
0 - Idle vs. 11 - Peer-to-Peer	< 0.0001	-1.66	< 0.0001	-3.30

Table 4.12: Hypotheses $H1$ Test Results

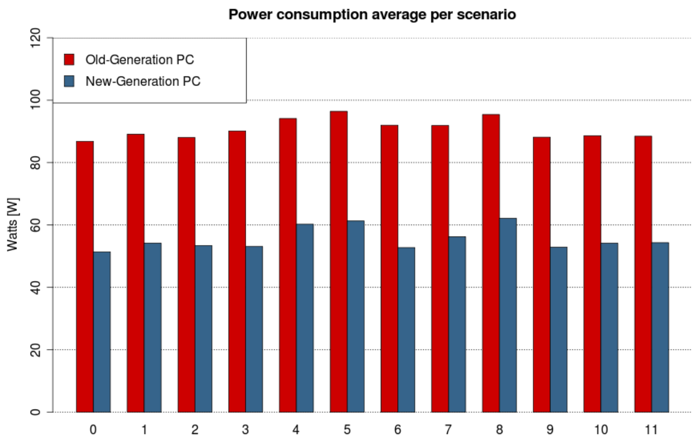

Figure 4.10. Per-scenario Power Consumption average values

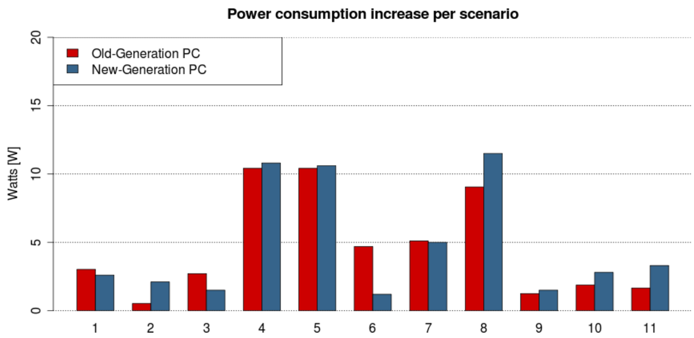

Figure 4.11. Per-scenario Power Consumption increase with respect to Idle

Question 2: Is it possible to classify software usage scenarios based upon power consumption?

$H2 : P_{idle} \neq P_{net} \neq P_{prod} \neq P_{file} \neq P_{MM}$

	Old-Gen PC		*New-Gen PC*	
Scenario Comparison	**p-value**	**Est. Diff**	**p-value**	**Est. Diff**
Idle vs. Network	< 0.0001	-2.08	< 0.0001	-3.20
Idle vs. Productivity	< 0.0001	-2.71	< 0.0001	-1.50
Idle vs. File System	< 0.0001	-10.41	< 0.0001	-10.60
Idle vs. Multimedia	< 0.0001	-1.67	< 0.0001	-1.60
Network vs. Productivity	< 0.0001	-0.31	< 0.0001	1.70
Network vs. File System	< 0.0001	-6.97	< 0.0001	-6.80
Network vs. Multimedia	< 0.0001	0.31	< 0.0001	1.60
Productivity vs. File System	< 0.0001	-6.87	< 0.0001	-9.10
Productivity vs. Multimedia	< 0.0001	0.73	< 0.0001	-0.20
File System vs. Multimedia	< 0.0001	8.53	< 0.0001	8.60

Table 4.13: Hypothesis $H2$ Test Results

Question 3: What is the relationship between usage and power consumption?

$H3_a :\neq max[\rho(I_{CPU}, P), \rho(I_{Memory}, P), \rho(I_{Disk}, P), \rho(I_{Network}, P)] > \beta$

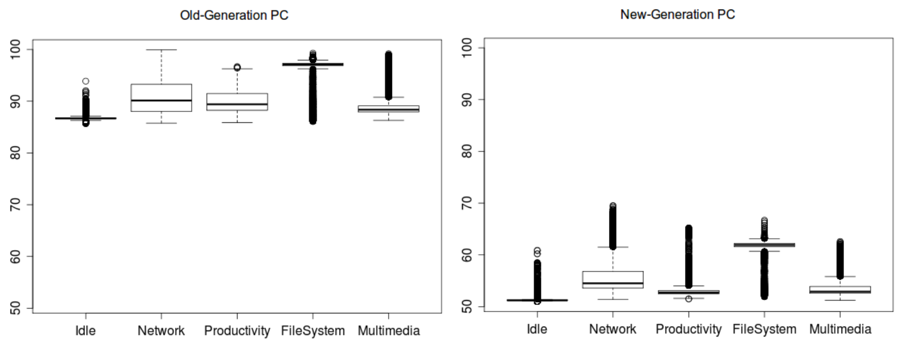

Figure 4.12. Box Plots of Scenario Categories

	Old-Gen PC		*New-Gen PC*	
Scenario	**Data Distr.**	**Max p-val.**	**Data Distr.**	**Max p-val.**
0 - Idle	Not Normal	1.5e-63	Not Normal	2.2e-39
1 - Web Navigation	Not Normal	4.4e-36	Not Normal	1.1e-20
2 - E-Mail	Not Normal	9e-73	Not Normal	1.2e-19
3 - Productivity Suite	Not Normal	1e-45	Not Normal	9.4e-29
4 - IO Operation (Disk)	Not Normal	1.2e-46	Not Normal	8.7e-51
5 - IO Operation (USB)	Not Normal	6.4e-52	Not Normal	2.5e-29
6 - Image Browsing	Not Normal	1.1e-35	Not Normal	6.7e-22
7 - Skype Call (No Video)	Not Normal	8.2e-30	Not Normal	3e-67
8 - Skype Call (Video)	Not Normal	1.3e-35	Not Normal	5.2e-36
9 - Multimedia Playback (Audio)	Not Normal	7.9e-54	Not Normal	5.2e-44
10 - Multimedia Playback (Video)	Not Normal	1.6e-44	Not Normal	6.6e-81
11 - Peer-to-Peer	Not Normal	8.9e-36	Not Normal	2.2e-35

Table 4.14: Data Distribution Analysis

Old-Generation PC				
Scenario Title	**Variable**	**p-value**	ρ	**R2**
2 - E-Mail	CPUC1Time.	< 0.0001	-0.36	13 %
4 - IO Operation (Disk)	CPUTime.	< 0.0001	0.35	12 %
4 - IO Operation (Disk)	CPUC1Time.	< 0.0001	-0.35	12 %
5 - IO Operation (USB)	CPUTime.	< 0.0001	0.47	22 %
5 - IO Operation (USB)	CPUC1Time.	< 0.0001	-0.47	22 %
7 - Skype Call (No Video)	CPUC1Time.	< 0.0001	-0.39	15 %
8 - Skype Call (Video)	CPUTime.	< 0.0001	0.63	40 %
8 - Skype Call (Video)	CPUUserTime.	< 0.0001	0.53	28 %
8 - Skype Call (Video)	CPUC1Time.	< 0.0001	-0.7	49 %
11 - Peer-to-Peer	MemoryKByteAvailable	< 0.0001	-0.34	12 %

Table 4.15: Spearman's ρ Coefficient between Power and Resource variables

New-Generation PC				
Scenario Title	**Variable**	**p-value**	ρ	**R2**
2 - E-Mail	CPUUserTime.	< 0.0001	0.42	17 %
2 - E-Mail	CPUPrivTime.	< 0.0001	0.43	18 %
3 - Productivity Suite	CPUUserTime.	< 0.0001	0.33	11 %
4 - IO Operation (Disk)	PhysicalDiskTransfers	< 0.0001	0.45	20 %
4 - IO Operation (Disk)	LogicalDiskTransfers	< 0.0001	0.45	20 %
4 - IO Operation (Disk)	MemoryPages	< 0.0001	0.44	19 %
4 - IO Operation (Disk)	MemoryKByteAvailable	< 0.0001	-0.54	29 %
4 - IO Operation (Disk)	CPUC3Time.	< 0.0001	-0.59	35 %
4 - IO Operation (Disk)	CPUTime.	< 0.0001	0.55	31 %
4 - IO Operation (Disk)	CPUUserTime.	< 0.0001	0.58	34 %
4 - IO Operation (Disk)	CPUPrivTime.	< 0.0001	0.39	15 %
6 - Image Browsing	CPUUserTime.	< 0.0001	0.34	12 %
7 - Skype Call (no video)	NetworkPkts	< 0.0001	0.62	39 %
7 - Skype Call (no video)	MemoryKByteAvailable	< 0.0001	-0.45	20 %
7 - Skype Call (no video)	CPUC3Time.	< 0.0001	-0.66	43 %
7 - Skype Call (no video)	CPUTime.	< 0.0001	0.52	27 %
7 - Skype Call (no video)	CPUUserTime.	< 0.0001	0.63	39 %
8 - Skype Call (Video)	NetworkPkts	< 0.0001	0.67	46 %
8 - Skype Call (Video)	MemoryKByteAvailable	< 0.0001	-0.62	39 %
8 - Skype Call (Video)	CPUC3Time.	< 0.0001	-0.88	77 %
8 - Skype Call (Video)	CPUTime.	< 0.0001	0.87	76 %
8 - Skype Call (Video)	CPUUserTime.	< 0.0001	0.9	81 %
9 - Multimedia Playback (Audio)	MemoryKByteAvailable	< 0.0001	-0.34	12 %
11 - Peer-to-peer	NetworkPkts	< 0.0001	0.45	20 %
11 - Peer-to-peer	MemoryKByteAvailable	< 0.0001	-0.42	18 %
11 - Peer-to-peer	CPUPrivTime.	< 0.0001	0.35	12 %

Table 4.16: Spearman's ρ Coefficient between Power and Resource variables

4.2.4 Discussion

The collected data shows several facts, and gives the answers for the Research Questions. As observed in Table 4.12, in both our test machines, every usage scenario consumes more power than the Idle scenario. This difference is even more evident in the New-Generation PC, where we obtain our highest increase percentage (up to 20%).

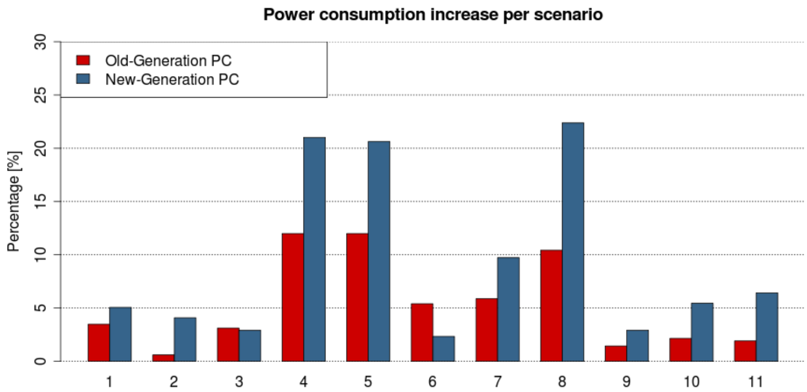

Figure 4.13. Per-scenario Power Consumption increase with respect to Idle (in %)

As regards RQ9, scenarios' classification, results are not homogeneous: for instance, in Figure 4.12 it can be observed that Network category has a very wide range if compared to the others. Moreover, the comparison does not always give a clear distinction between the profiles. This suggests that a classification based on functionality can be inadequate for power consumption. Another classification may arise from the analysis of every single scenario. As can be seen from Tables 4.10, 4.11 and 4.12, the most power-consuming scenarios are those that involve File System, followed by Skype (both with and without Video Enabled) and Image Browsing. From the hardware point of view, these scenarios are also the most expensive in terms of system resources. Thus, classifying our scenarios based upon resource utilization can be a more accurate way to estimate their power consumption. For instance, the power consumption profile of Skype is very different (about 4-5 Watts in average) with and without enabling the Video Camera.

Another interesting question that arises from the analysis is, in the case of applying these Scenarios in groups, if their power consumption would follow a linear composition rule (thus summing up the values). That is, for example, supposing a composed Usage Scenario S that involves a Skype Call, a Web Navigation and a Disk Operation performed simultaneously, their linear composition would give, on our Old-Gen PC, an estimated Power Consumption per second of

$$P_{idle} + \Delta P_S = 86.81W + 21.33W = 108.14W$$

introducing a 25% overhead on power consumption. On the New-Gen PC, the estimated Power Consumption would be

$$P_{idle} + \Delta P_S = 51.39W + 24.90W = 76.29W$$

which gives a 48% overhead on power consumption.

Taking a look at the results of the correlation analysis, represented by RQ 10, more conclusions can be made. First of all, we can observe that the coefficients related to the New-Gen PC are higher with respect to the Old-Gen PC. This may suggest that as hardware evolves, the software usage is even more significant for determining the power consumption of the system. This assumption is confirmed by Figure 4.13, where we can observe that the percentage increase of the New-Gen PC is higher, in most cases, with respect to the Old-Gen.

However, it is remarkable that, for both machines, the variables that show higher correlation coefficients are undoubtedly those related to CPU Usage and Memory Usage. High coefficients are also present in the Hard Disk Index, but only in those scenarios that, clearly, involve File System operations. This means that those resources have a greater influence upon power consumption related to the others selected for the analysis. Further researches should probably focus upon these two variables.

AAs expected, power consumption always has a *negative* correlation with the time spent by the CPU in the C1 and C3 states, which are power saving, low-activity states, and with the available memory, which means that using more memory has a positive correlation with power, which is a reasonable and correct behaviour. This is also a confirmation that the analysis was conducted with the right premises.

Moreover, as expected, the scenarios which exhibit higher correlations are those which use more resources, such as Skype and IO scenarios. In particular, the Skype scenario with video enabled has a strong correlation with the CPU usage, probably because the real-time video elaboration makes the CPU the dominant resource for power consumption.

4.2.5 Considerations and Future Work

This experiment assessed quantitatively the energetic impact of software usage. It consisted of building up common application usage scenarios (e.g.: Skype call, Web Navigation, Word writing) and executing them independently to collect power consumption data. Each single scenario introduced an overhead on power consumption, which may raise up to 20% for recent systems: if their power consumption would follow a linear composition rule, the impact could be even higher.

The relationship between usage and power consumption was also analysed in terms of correlation between resource usage. Although a clear linear relationship did not arise, the analysis showed that some resources drive power consumption more than others, such as memory and CPU usage. Using a precise control over how an application consumes these resources, it can be possible to predict its power consumption, thus including dedicated countermeasures in the Software Design Phase – which, by itself, is the essence of Energy-Aware Programming.

Our experiment also gives us the indication that modern Desktop systems, although being more energy efficient in standby and idle states, due to their higher scalability, are even more sensitive to the impact of software usage on power consumption. This indicates that research should focus on reducing this impact, as it will always be more significant as time goes by.

Moreover, results set the basis for future work and research projects. A more accurate correlation analysis will be conducted, focusing on the more relevant resources and also taking into account different kinds of relationships (not just linear). Moreover, we will focus our attention on battery-powered mobile devices, where software power consumption is a key issue. Our idea is that re-factoring applications by considering a more efficient resource utilisation, the impact of software on power consumption could easily be reduced.

4.3 Mobile

The energy profiling of mobile devices is an active research stream, especially as regards mobile and embedded devices. The concept of energy-awareness is based upon a complete knowledge on how and where energy is consumed on a device. In [20], authors present a detailed analysis of power consumption in a mobile device, focusing on the hardware subsystems, through common and realistic usage scenarios. Results show that the GSM module and the display are the most power-consuming components: for example, a GSM phone call on OpenMoko Neo Freerunner, HTC Dream G1 and Google Nexus One consumes 1135 mW, 822 mW and 846 mW respectively.

Usually, an accurate power consumption analysis of mobile or embedded devices is component-based. However, instantaneous information about discharge current and remaining battery capacity is not always available, because most devices do not have built-in sensors to collect these data. In [108], a technique called PowerBooter is proposed to build a battery-based model automatically. Authors motivate this decision by considering that different mobile devices of the same category show different power consumption, and a specific power consumption model for each device is difficult to obtain. Thus, instead of using external metering instrumentation to detect power consumption, only the internal battery voltage sensor is used, which is found across many modern smartphones. Authors indicate that for a 10-second interval, the PowerBooter technique has an accuracy of about 4.1% within measured values.

From a software engineering point of view, most contributions are devoted to developing frameworks and tools for energy metering and profiling. Also in [108], authors propose an on-line power estimation tool called PowerTutor. It implements the PowerBooter model in order to profile power consumption of applications, based upon their component usage. Another example, which makes use of external metering devices, is ANEPROF [25], which authors define as a real-measurement-based energy profiler able to reach function-level granularity. It is developed for Android OS-based devices, thus it is aimed at profiling Java applications. It is based on JVM event profiling, using software probes to record runtime events and system calls. Authors had to address several design issues, such as overhead control and proper time synchronization. Power consumption profiling is made through correlation of real-time power measurements done by an external DAQ, connected to a ARM Computer-on-Module running Android 2.0. Authors also provide profiling data of four popular applications (Android Browser, GMail, Facebook, Youtube). The accuracy of ANEPROF depends on the hardware meter used. Its CPU overhead is stated to be less than 5%. Finally, SEMO [28] is a smart energy monitoring system, developed for Android, which also provides application-level consumption monitoring. This system is composed of three components: a *inspector*, which monitors the information on the battery, warning users when the battery reaches a critical condition; a *recorder*, which basically logs the actual charge of the battery and the running applications, and a *analyser*, which calculates the energy consumption rate for each application and ranks them according to it.

As we have shown in this section, several efforts have been made as regards energy profiling in mobile devices. However, these works differ greatly in terms of methodologies and formalisms used. Palit et al. in [78] propose an interesting framework for performing experiments to measure the energy cost of software applications on smartphones. They define the concept of *user-level test case* γ_i as a pair $< input; output >$ where the input is composed of an application setting α_i and a device configuration β_i, and the output is the energy cost θ_i, expressed as a custom metric depending on battery capacity and amount of current consumed. Formally: $\gamma_i =< \alpha_i, \beta_i; \theta_i >$
Authors also describe a typical workbench for experimentation, which is very similar to the one we used in this work.

4.3.1 Study Design

The aim of our research [10] is to compare the impact of software usage on power consumption in two different Android OS-based mobile phones. For this purpose, we performed two experiments: the first one, called "training tools", fixes a set of high level features in order to compare the power consumption between the two devices, while the second one, called "gLCB", executes different profiles of a Context-Aware application and compares its power consumption on the two devices according to the selected profiles.

4.3.2 Goal Description and Research Questions

We define our goal through the Goal-Question-Metric (GQM) approach. [99]. This approach, applied to our experiment, lead to the definition of the model presented in table 4.17.

Goal 1	Analyse usage scenarios of two mobile devices for the purpose of assessing differences with respect to power consumption from the viewpoint of the System User in the context of mobile applications
Question 11.1	Do usage scenarios have the same power consumption?
Metric	P_i, j Power consumed by the scenario i by device j; $i[0, 10], j$[1,2]
Question 11.2	Is the energy consumption the same among the devices?
Metric	P_i, j Power consumed in every scenario i by device j; $i[0, 10], j$[1,2]
Goal 2	Analyse usage profiles of gLCB source code for the purpose of assessing differences with respect to power consumption from the viewpoint of the System User in the context of mobile applications
Question 12.1	Does gLCB cause a variation of the devices power consumption?
Metric	Power (Watt)
Question 12.2	Are there statistical differences between different user profiles?
Metric	Power (Watt)
Question 12.3	Are there statistical differences between the behaviour of gLCB in different devices?
Metric	Power (Watt)

Table 4.17: The GQM Model for our experiments

4.3.3 Variable selection

Experiment 1: Training Tools. We aim at quantifying, in two different models of smartphone, the power consumption of hardware components, when performing daily activities for a common user. We selected two independent variables: the smartphone model (M) and the specific scenario (S). Each has been executed 30 times, with a fixed duration of 4 minutes per scenario. Our dependent variable is the consumed power (P).
S0: Standby. This scenario provides the baseline for our analysis. During this scenario, there are no user applications in execution, and 2G and 3G connections are enabled.

S1: Phone call over 2G network. This scenario executes a phone call to a prefixed number, for a duration of 4 minutes.
S2: Phone call over 3G network. Same as above, except that the call is made using the UMTS network.
S3: File download through WiFi connection. In this scenario, the scheduled task launches a new thread, which downloads a remote file, the Ubuntu 11.10 disk image, up until the scheduled timeout (4 minutes).
S4: File download through 2G (EDGE/GPRS) connection. Same as above, except that the downloaded file is the Android SDK, which is smaller in size.
S5: File download through 3G (UMTS) connection. Same as above, except that the UMTS network is used.
S6: Localization activity through GPS. This scenario manages position updates. The task simply registers on location updates and reads the new values of latitude and longitude, up to the 4 minutes timeout.
S7: Scan for Bluetooth devices. In this scenario, a scan for Bluetooth devices is performed. The scan process lasts, according to specifications, 12 seconds on average. At the end of the scan procedure, the task simply restarts, up until the prefixed duration.
S8: CPU-intensive activity. The aim of this scenario is maintaining a high CPU workload while gathering power consumption data. For this purpose, repeated cryptography operations are performed, with a pool of 20 threads, each of them iterating the procedure 10 times.
S9: Playback of an audio file. This scenario plays an mp3 compressed audio file, 4.78 MB in size, played in loop up until the scheduled timeout.
S10: Active display with 50% Brightness. The aim of this scenario is assessing the impact of the active display over power consumption. This scenario is similar to S0, the only difference being that all radios (2G, 3G, WiFi) and the SIM card were disabled.

Experiment 2: gLCB. We analyse the energetic behaviour of an application, called gLCB, the Android porting of a Context-Awareness[5] application developed by Telecom Italia Lab, which will be described in detail in section 4.5. Basically, its purpose is to retrieve diverse context information (such as geographical location, WiFi hotspots, Bluetooth devices, etc.) from a portable device, in order to send it to a remote Context Provider for the implementation of Context-Aware services to the end user. gLCB is based on an event mechanism, that triggers the data upload only when a context change is detected. Depending on the usage profile chosen for

[5] Telecom Italia, Context Awareness: servizi mobili su misura, `http://www.telecomitalia.it/TIPortale/docs/innovazione/012007/Pag11-22%20contextawareness.pdf`

the application, which can be one of the following: *VERY LOW, LOW, NORMAL, HIGH, AUTO, CUSTOM*, the data retrieving and upload ratio are adjusted, thus affecting the energy behaviour of the application. In our experiment, each profile was set through a server application and data was collected during execution sessions of the gLCB application of the fixed duration of 60 minutes for each profile.

Hypothesis Formulation

Based upon our GQM Model, we can formalise our Research Questions into Hypotheses.

Experiment 1: Training Tools

- *RQ 11.1: Do usage scenarios have the same power consumption?*
  $H_{0,1}$: $P_{0,1} = P_{1,1} = ... = P_{10,1}$
  $H_{0,2}$: $P_{0,2} = P_{1,2} = ... = P_{10,2}$

  $H_{A,1}$: $P_{0,1} \neq P_{1,1} \neq ... \neq P_{10,1}$
  $H_{A,2}$: $P_{0,2} \neq P_{1,2} \neq ... \neq P_{10,2}$

- *RQ 11.2: Is the energy consumption the same among the devices?*
  H_0: $P_{i,1} = P_{i,2}$, $i \in [0, 10]$
  H_A: $P_{i,1} \neq P_{i,2}$, $i \in [0, 10]$

Experiment 2: gLCB

- *RQ 12.1: Does gLCB cause a variation of the devices power consumption?*
  $H_{0,1}$: $P_{1 \text{ with gLCB}} = P_{1 \text{ without gLCB}}$
  $H_{0,2}$: $P_{2 \text{ with gLCB}} = P_{2 \text{ without gLCB}}$

  $H_{A,1}$: $P_{1 \text{ with gLCB}} \neq P_{1 \text{ without gLCB}}$
  $H_{A,2}$: $P_{2 \text{ with gLCB}} \neq P_{2 \text{ without gLCB}}$

- *RQ 12.2: Are there statistical differences between different user profiles?*
  $H_{0,1}$: $P_{1 \text{ high}} = P_{1 \text{ normal}} = P_{1 \text{ low}} = P_{1 \text{ verylow}}$
  $H_{0,2}$: $P_{2 \text{ high}} = P_{2 \text{ normal}} = P_{2 \text{ low}} = P_{2 \text{ verylow}}$

$H_{A,1}$: $P_{1\ high} \neq P_{1\ normal} \neq P_{1\ low} \neq P_{1\ verylow}$
$H_{A,2}$: $P_{2\ high} \neq P_{2\ normal} \neq P_{2\ low} \neq P_{2\ verylow}$

- *RQ 12.3: Are there statistical differences between the behaviour of gLCB in different devices?*
  H_0: $P_{i,1} = P_{i,2}$, $i \in$ (high, normal, low, verylow)
  H_A: $P_{i,1} \neq P_{i,2}$, $i \in$ (high, normal, low, verylow)

Instrumentation and Experiment Design

The selected usage scenarios have been implemented in Java code using the Android SDK. In order to obtain a statistically relevant data set, each scenario has a fixed execution time of 4 minutes, and each execution was repeated 30 times. This procedure was equally applied on each smartphone.

Hardware Instrumentation. The experiments were performed on two different models of smartphones: the "Galaxy i7500", first announced in April, 2009, which is the first model produced by Samsung based on Android OS; and the "Nexus S", first announced in December, 2010, produced by Google and Samsung. Their technical specifications are listed in the producer website[6].

The power consumption data was acquired through a power metering architecture. The battery was removed from the devices, in order to avoid bias due to discharge and subsequent OS power saving procedures. The battery terminals were directly connected to a DC power supply, providing 5 V steadily. This value was chosen after different tests, that showed how lower values were not able to maintan the device operational during the most of the power consuming scenarios, because of the voltage drop on the shunt resistance. The DC power supply used is the TPS-2000D produced by Topward Electric Instruments Co. A Data Acquisition Board (DAQ), the DAQLite produced by Eagle Technology, was used to acquire the power consumption data. The DAQ was set to a sampling frequency of 350Hz, in order to produce an amount of data statistically relevant, but not prohibitive for subsequent computation.

Software Setup. In order to automate scenario execution in our experiments, a supporting software environment was developed, composed of two Android applications, a server-side application and macro scripts, to be executed by the tool AutoHotKey[7]. The developed Android application allows the enabling or disabling

[6] `http://www.samsung.com/`

[7] AutoHotKey, a macro open-source utility , `http://www.autohotkey.com`

components, such as Bluetooth, GPS or WiFi interface, in order to avoid bias during scenarios that do not use them. For our second experiment, another Android application has been developed to control the execution of gLCB, specifying an execution time and a usage profile. With this solution, we assessed how the execution of different profiles of the application affected the power consumption of the device. These applications communicate with a server machine, which is then connected to the DAQ via USB. The server application then launches a AutoHotKey script that performs the needed operations for data acquisition and logging.

Analysis methodology

The goal of data analysis is to apply appropriate statistical tests to reject the null hypothesis. As we expected, the collected power consumption values, for both smartphones, do not follow normal distribution. This was verified by means of the Shapiro-Wilk test, with a resulting p-value lower than 0.05. This is true for our first experiment, "Training tools" as well as for the second one, "gLCB". Thus, in order to verify our hypotheses, we used non-parametric versions of the Kruskal-Wallis and Wilcoxon-Mann-Whitney tests, to assess the statistical independence between the different scenarios and profiles evaluated during our experiments. Again, we will draw conclusions from our tests based on a significance level $\alpha = 0.05$, that is we accept a 5% risk of type I error – i.e. rejecting the null hypothesis when it is actually true.

Threats to validity

We will classify threats of experiment validity into two categories: **internal** threats, derived from our treatments and instrumentation, and **external** threats, regarding the generalization of our work. A possible internal threat concerns the sampling frequency adopted by the DAQ, namely 350 Hz. We chose this frequency value for practical reasons, in order not to obtain a huge amount of data which could not be computed in a reasonable time by our servers. However, this frequency, compared to the operational frequencies of the selected smartphones, could be seen as quite low. A more significant threat comes from the usage, in some of our scenarios, of different communication networks which are characterised by an unpredictable behaviour. This behaviour may add bias to our measurements, introducing a high data variability. For example, as regards the cellular network, power consumption could be affected by the following mechanism: the base station to which the mobile device is connected detects the signal power, and if the *SINR* (Signal-to-Interference Plus Noise Ratio) is below or above a specific threshold it may negotiate a signal power increase or reduction to the device antenna. Finally, although is it not possible to generalize our results, because we performed our experiments on two specific models of smartphones, it is however possible to consider them as representatives of a category of devices with similar specifications.

4.3.4 Preliminary Data Analysis

We present in tables 4.18, 4.19, 4.20, 4.21 the following descriptive statistics about collected data. Tables report in this order: mean (milliWatts), median (milliWatts), standard deviation (σ), variation coefficient (the standard deviation divided by the mean). Tables 4.18, 4.19 contain descriptive statistics for each scenario of our "Training Tools" experiment, while tables 4.20, 4.21 contain descriptive statistics for each profile of our "gLCB" experiment.

Scenario	**Median(mW)**	**Mean(mW)**	**Std.Dev.**	**Var.Co.**
2G Standby	8.663	17.840	65.763	3.686
3G Standby	8.663	27.248	97.592	3.581
2G Call	658.618	746.447	371.118	0.497
3G Call	957.803	988.069	97.313	0.098
WiFi Download	628.724	646.604	61.403	0.094
2G Download	669.467	784.099	742.696	0.947
3G Download	955.175	947.515	181.155	0.191
GPS	450.189	484.753	79.748	0.164
Bluetooth Scan	251.526	273.960	78.018	0.284
CPU-Intensive	606.923	608.708	38.442	0.063
Mp3 Audio	324.720	374.971	142.073	0.378
Display	386.252	408.754	81.598	0.199

Table 4.18: "Training Tools": Scenarios Statistics - Galaxy i7500

Scenario	**Median(mW)**	**Mean(mW)**	**Std.Dev.**	**Var.Co.**
2G Standby	8.663	26.830	55.572	2.071
3G Standby	8.663	18.958	48.708	2.569
2G Call	379.488	543.487	565.230	1.040
3G Call	846.688	878.850	126.708	0.144
WiFi Download	455.733	513.046	166.444	0.324
2G Download	605.874	722.422	854.086	1.182
3G Download	965.368	931.798	208.819	0.224
GPS	296.626	300.444	20.375	0.067
Bluetooth Scan	217.571	227.051	42.882	0.188
CPU Intensive	886.552	877.747	54.055	0.061
Mp3 Audio	155.035	164.709	26.666	0.161
Display	598.734	708.075	177.169	0.250

Table 4.19: "Training Tools": Scenarios Statistics - Nexus S

Profile	Median(mW)	Mean(mW)	Std.Dev.	Var.Co.
Very-low	396.474	521.565	213.243	0.408
Low	396.474	544.451	226.976	0.416
Normal	455.869	594.340	248.253	0.417
High	514.952	617.016	253.448	0.410

Table 4.20: "gLCB": Profile Statistics - Galaxy i7500

Profile	Median(mW)	Mean(mW)	Std.Dev.	Var.Co.
Very-low	420.271	536.863	225.055	0.419
Low	435.116	556.254	244.286	0.439
Normal	690.392	834.233	315.046	0.377
High	808.711	876.160	334.903	0.382

Table 4.21: "gLCB": Profile Statistics - Nexus S

Hypothesis Testing

In this section we provide the results of hypothesis testing for our research questions. All p-values have been verified to be lower than the chosen significance level $\alpha = 0.05$. For further details, full values are available on our group website[8].

- *Research Question 11.1* - Do usage scenarios have the same power consumption?
  - $H_{1,1} : P_{0,1} \neq P_{1,1} \neq ... \neq P_{10,1}$
    Our values range from an average of 17.8 mW for Scenario S0 to an average of 988 mW for Scenario S2. The Kruskal-Wallis test for the hypothesis resulted in a p-value lower than 2.2e-16. Thus, we reject the null hypothesis.
  - $H_{1,2} : P_{0,2} \neq P_{1,2} \neq ... \neq P_{10,2}$
    Our values range from an average of 18 mW for Scenario S0 to an average of 931.8 mW for Scenario S5. The Kruskal-Wallis test for the hypothesis resulted in a p-value lower than 2.2e-16. Thus, we reject the null hypothesis.

- *Research Question 11.2* - Is the energy consumption the same among the devices?

[8] `http://softeng.polito.it/pvalues.pdf`

- H_2: $P_{i,1} \neq P_{i,2}$, $i \in [0, 10]$
  The Mann-Whitney test resulted in a p-value lower than 0.001 for each scenario. Thus, we reject the null hypothesis.

- *Research Question 12.1* - Does gLCB cause a variation of the devices power consumption?
  - $H_{1,1}$: $P_{1\ \text{with gLCB}} \neq P_{1\ \text{without gLCB}}$
    The Mann-Whitney test resulted in a p-value lower than 2.4e-09 for each profile compared to the standby consumption. Thus, we reject the null hypothesis.
  - $H_{1,2}$: $P_{2\ \text{with gLCB}} \neq P_{2\ \text{without gLCB}}$
    The Mann-Whitney test resulted in a p-value lower than 1.5e-09 for each profile compared to the standby consumption. Thus, we reject the null hypothesis.

- *Research Question 12.2* - Are there statistical differences between different user profiles?
  - $H_{2,1}$: $P_{1\ \text{high}} \neq P_{1\ \text{normal}} \neq P_{1\ \text{low}} \neq P_{1\ \text{verylow}}$
    Our values range from an average of 521.5 mW for *Very Low* profile to an average of 617 mW for *High* profile. The Kruskal-Wallis test for the hypothesis resulted in a p-value lower than 1.146e-15. Thus, we reject the null hypothesis.
  - $H_{2,2}$: $P_{2\ \text{high}} \neq P_{2\ \text{normal}} \neq P_{2\ \text{low}} \neq P_{2\ \text{verylow}}$
    Our values range from an average of 536.8 mW for *Very Low* profile to an average of 876 mW for *High* profile. The Kruskal-Wallis test for the hypothesis resulted in a p-value lower than 3.433e-15. Thus, we reject the null hypothesis.

- *Research Question 12.3* - Are there statistical differences between the behaviour of gLCB in different devices?
  - H_3: $P_{i,1} \neq P_{i,2}$, $i \in$ (high, normal, low, verylow)
    The Mann-Whitney test resulted in a p-value lower than 2e-10 for each profile. Thus, we reject the null hypothesis.

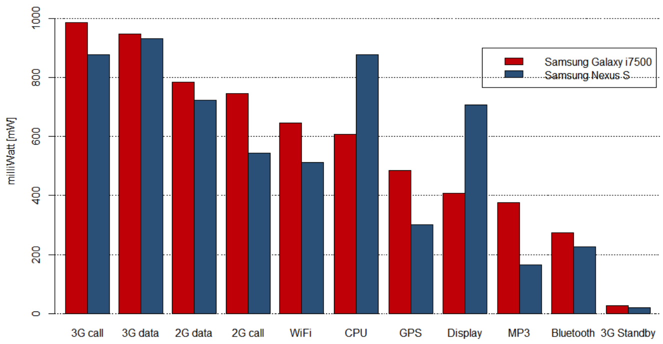

Figure 4.14. Instant Power Consumption (avg values) comparison

The barplot in figure 4.14 shows the average power consumption values in mW for each scenario, on both smartphone models. As we expected, we may notice that the standby values are the lowest, below 27 mW in the worst case. From the graph, it is evident how the most recent smartphone, the Samsung Nexus S, consumes a significantly lower amount of power in each scenario, an exception given by the CPU-Intensive and the Active Display scenarios. The percentage variations between the two smartphones, reported in table 4.22, spread from a minimum -56,08% in the Mp3 Audio scenario, to a +73,23% in the Active Display scenario.

Scenario	**Samsung Galaxy**	**Samsung Nexus S**	**Galaxy vs Nexus**
Display	408,75	708,08	+73,23%(299,33 mW)
CPU Intensive	608.71	877,75	+44,19% (269 mW)
3G Download	947,51	931,80	-1,65% (15,7 mW)
2G Download	784,10	722,42	-7,86% (61,6 mW)
3G Call	988,06	878,85	-11,05% (109 mW)
Bluetooth Scan	273,96	227,05	-17,12% (46,9 mW)
WiFi	646,60	513,05	-20,65% (133,5 mW)
2G Call	746,45	543,49	-27,19% (202,9 mW)
GPS	484,75	300,45	-38,02% (184,3 mW)
Mp3 Audio	374,98	164,70	-56,08% (210,2 mW)

Table 4.22: Smartphones power consumption comparison

As regards our gLCB experiment, the barplot in figure 4.15 shows the average power consumption values in mW for each profile, on both smartphone models. It is possible to notice that, between the two, the most recent smartphone consumes less in *verylow* and *low* profiles.

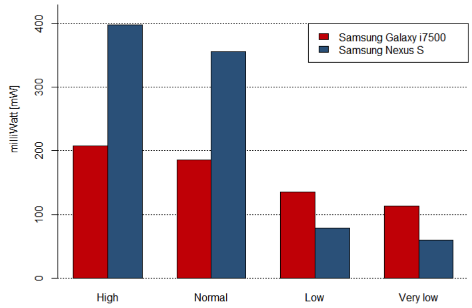

Figure 4.15. Instant Power Consumption (avg values) of gLCB energy profiles

4.3.5 Discussion

Experiment 1: "Training Tools"
From the results obtained from our first experiment, we can conclude that the most power consuming user activities, among the ones we selected, on both the smartphone models used, are those that use the radio module, namely the phone call and data transfer on both 2G (EDGE/GPRS) and 3G (UMTS) networks. This finding is coherent with [20], in which the most power consuming scenario is indeed the phone call. Also in [78], examining some of the network-related scenarios, it emerges that the battery lasts longer in all cases if the WiFi network interface is used, rather than 2G or 3G. From our results, it is worth noticing how 3G network causes a sensible increase in power consumption with respect to 2G, in both voice and data communications. Moreover, as regards the power consumption difference between the two smartphone models, we notice that the most recent model has in general a lower power consumption, the only exception being the CPU-intensive and active display scenarios: this can be justified, considering the increase in CPU frequency (1 GHz compared to 528 MHz) and in display dimensions (4" compared to 3.2") which characterise the most recent model.

Experiment 2: "gLCB"
As regards our gLCB experiment, it is immediately noticeable that in every profile the Samsung Nexus S consumes a higher amount of power. This is likely because during this experiment, the display of the smartphone was active, in order to verify the correct execution of the application, and the switching between profiles and execution sessions. This was done on purpose, in order not to introduce random bias due to occasional checking of the application behaviour. Instead, we may subtract the display overhead from the power consumption values, and this is valid because we

know that, from the previous experiment, the display consumption is characterised by low variance and dispersion values. The recalculated values, without the display overhead, are shown in table 4.23. It is interesting to notice how, from these results, it emerges that the Nexus S actually has lower power consumption values than the Galaxy in profiles *verylow* and *low*, namely 47,5% and 42,1% respectively. The other profiles show a significantly higher power consumption. Given that the step-up from *low* to *normal* profile is characterized by the activation of WiFi, Bluetooth and GPS components, during *normal* and *high* profiles a higher computational load is expected. Thus, we may conclude that the power consumption increase is due to the CPU activating more frequently than in the other two profiles, also because we know, from the results of the previous experiment, that the CPU has a higher impact on the Nexus S smartphone. These results show that the impact of the gLCB application in terms of power consumption gradually reduces, by adopting lower energy profiles, on both smartphones.

Profile	**Samsung Galaxy (mW)**	**Samsung Nexus S (mW)**
Very-low	112.811	59.185
Low	135.697	78.576
Normal	185.586	356.555
High	208.262	398.482

Table 4.23: Smartphones power consumption comparison (no display overhead)

4.3.6 Considerations and future work

From the analysis of the results provided by our experiments, we can conclude that the most recent device, in terms of OS and hardware components, shows significantly lower power consumptions than the least recent one, except for the CPU-intensive and active display cases. This proves that technology improvements have a great impact over power consumption reduction. Moreover, we showed that different execution profiles of the same application can significantly affect the power consumption of a device. This finding proves that energy-aware software applications can greatly improve the energy efficiency of mobile devices, while providing the same functionalities.

As regards future work, it would be interesting to profile the energy consumption of other usage scenarios, for example those who require a higher interaction between the user and the device. Moreover, because of our experiment design, the smartphones were constrained to a single physical location; it could be interesting profiling the power consumption of a moving user, in order to get closer to the real case and evaluate with more precision the contribute of the subsequent handoffs between different cells in mobile networks. Another interesting point of view could be analysing the power consumption of different generations of smartphones running the same version of the Android OS, in order to isolate the only impact of hardware changes. The issue of sustainability is starting to be addressed among the software engineering community. Although, during the First International Workshop on Green and Sustainable Software[9], there was common agreement that sustainability is and will be a key aspect of software, it is still unclear how to design sustainable software. While for other characteristics (reliability, performance, security, etc.) processes and metrics have been proposed and widely investigated by the SE community, as regards sustainability the discussion is still in its initial phase. In addition to that, software sustainability has an intrinsic difficulty because the topic invests not only technological aspects, but also economic, social and environmental, which are under the broad umbrella of sustainability as defined in 1987 by the Bruntland commission[10]. Among the kaleidoscope of aspects related to software sustainability, one of the most visible is the energy (or, alternatively, power) consumption of software systems. Indeed, software does not consume energy directly; however it has a direct influence on the energy consumption of the hardware underneath. In fact, applications and operating systems indicate how the information is processed and, consequently, drive the hardware behaviour: as described in chapter 4, software can increase the total power consumption of a computer system up to 10%. This and other initial findings open investigation spaces on the optimization of energy and power consumption of IT devices acting on the software instead of the hardware. Moreover, nowadays the same software runs on multiple devices, thus it might be more productive and feasible for software houses to green the single software rather than relying on the greening of all the hardware implementations underneath (that could require competences commonly not owned by software houses). Optimizing a software product in terms of energy efficiency also has some issues. The absence of a standard procedure, or a benchmark, to compare systems is the most prominent one. This is because software is intangible and it is deployed on devices with their own specifications and features. This makes it really difficult to standardise a transparent, platform-independent measuring system for every software system. Another consideration must be done regarding software architectures. During recent years, software engineers always tried to increase the number of software layers - that is, for improving interoperability, abstraction, decoupling, etc. However, the steep increase of software layers directed the optimisation efforts only on each layer ("horizontal" optimisation) and not across them ("vertical" optimisation). Since energy efficiency

[9] http://greens.cs.vu.nl/, last visited on September 13th, 2012

[10] In 1987 the Brundtland commission defined sustainable development as "a development that meets the needs of the present, without compromising the ability to meet the needs of the future [18]"

directly relates with hardware technologies, a more intense communication flow between hardware and software is needed to achieve significant optimisations. In this sense, embedded systems make a perfect case study, because their architecture is simplified by design, and also because power consumption issues acquire a peculiar importance, for operational reasons (most embedded systems are battery-powered). For this reason our work uses an embedded system as the testbed to validate a new approach for the design and implementation of sustainable software. We investigate, and here we also introduce the goal and main contribution of this study; how software can be optimised by identifying code patterns that use in a sub-optimal way the hardware resources. These code patterns ought to be refactored in order to improve the energy efficiency of the software at run time. We define and name the code patterns Energy Code Smells, inspired by the well-known book of Fowler and Beck [33]. This study empirically validates the impact of Energy Code Smells over power consumption [101]. In Section 4.4, we provide some background and we give our definition of Energy Code Smells, then in Section 4.4.1 we describe the used approach for the validation of the concept. In Section 4.4.3 we describe the experimental setup of our analysis. In Sections 4.4.5, 4.4.6 and 4.4.6 we present and discuss our experimental results, along with the threats to validity that might affect our research. Finally we expose (Section 4.4.7) our conclusions and future research.

4.4 Energy Code Smells: background and definition

The term "code smells" was coined by Fowler and Beck [33] referring to poor implementation choices that make the software difficult to maintain. These bad implementation practices can be characterised as patterns in source code. For instance, the smell "*Long Method*" refers to a method that has grown too large: typically, the longer the method is, the more difficult it is to maintain it. One or more refactoring actions are associated to code smells: for example, all you have to do to refactor a Long Method is to extract parts of the method that seem to go nicely together and make a new method. As a result the original method is shorter and easier to maintain. The goal of identifying and refactoring code smells is to make the code more understandable and flexible to evolution, i.e. more maintainable, and many studies in the literature have been devoted to this aspect [107] [55]. However refactoring code smells might also have an effect on other properties of the software, such as the portability, the testability or, as in the case of this work, the energy efficiency of the code. As a consequence, we take inspiration from the original work of Fowler and Beck and we introduce the concept of smells into the Green IT community, introducing the Energy code Smells:

An Energy Code Smell is an implementation choice that makes the software execution less energy efficient.
Since software has different levels of abstractions and organisations, Energy Code Smells can be located at code, design or architectural level. Therefore, Energy Code Smells are implementation choices *at source code level* (code patterns) that make a sub-optimal usage of the hardware resources underneath. As a consequence, they provoke a higher energy (or alternatively, power) consumption.

4.4.1 Validation of Energy Code Smells

The aim of our research is to identify Energy Code Smells. In addition to that, we are also interested in understanding whether the Energy Code Smells also degrade the performances of the application in terms of execution time. We set up two research questions for our investigation:

RQ13. Which code patterns have an effect on power consumption (i.e., which code patterns are Energy Code Smells)?

RQ14. Do code smells that have an effect on execution time also have an effect on energy consumption (i.e. are Energy Code Smells also Performance Smells)?

The epistemological approach adopted for this research is the empirical one. We set up an experiment observing two dependent variables: power consumption (W) for RQ13 and execution time (ms) for RQ14. The two dependent variables are measured on the execution of C++ functions running on an embedded device. The choice of the embedded device has several advantages, the main two being:

- it has no operating system and thus confounding factors in the experiment are minimised;

- it runs on a battery and it really needs energy-efficient code.

In other terms, refactoring Energy Code Smells in such an environment might lengthen the life of the battery. The potential Energy Code Smells selected for the experiment are code patterns used by two popular static analysis tools. For each code pattern selected for the experiment, we set up a C++ function with two implementations, one that violates the code pattern (thus contains a Code Smell) and the refactored one without the violation. Therefore the treatment is the refactoring of the smell and it is possible to observe an effect on the two variables by comparing the measurements on the two versions of the code. Figure 4.16 represents the design described.

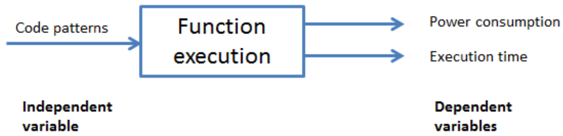

Figure 4.16. Experiment design.

4.4.2 Potential Energy Code Smells selection

As introduced above, the software that runs on the selected device is C++ code. In order to identify Energy Code Smells on C++ code we look at already existing code patterns. In particular, we examined patterns implemented by Automatic Static Analysis (ASA) tools. ASA tools examine source and compiled code and check it against good programming practices and possible bug patterns. The advantage of using ASA tools is the speed of the verification and the applicability before testing or production phase. The two tools selected for this study are Cpp-Check[11] and Findbugs[12]. CppCheck is a well-known static analysis tool for C/C++, which contains many patterns regarding a variety of desired software properties: safety, portability, performance, etc . An example of C/C++ pattern on portability is "*64 bits portability*", i.e. assign address to int or long. An example of checked pattern on performance is instead "*Address not taken*" of the category "Memory leaks", which detects when the address to allocated memory is not taken. In order to identify which patterns can be considered relevant for energy efficiency, two of the authors carefully read all patterns and selected independently which ones could cause a higher power consumption of the Waspmote. All conflicts (a pattern selected by only one expert) were resolved in a reconciliation meeting, where patterns were discussed and a final decision taken. In addition to the Cpp-Check patterns, we also reviewed the patterns of another static analysis tool, Findbugs. It is similar to Cpp-Check, but it analyses Java code. The same two authors reviewed all FindBugs patterns and decided firstly if they can be applied to C++ code, then whether they might be related to energy efficiency. The selection process ended up with the patterns shown in TABLE 4.24.

[11] `http://cppcheck.sourceforge.net/`, last visited on September 13th, 2012

[12] `http://findbugs.sourceforge.net/`, last visited on September 13th, 2012

Pattern Name	**Pattern Description**	**Tool**
Parameter By Value	Passing a parameter by value to a function	CppCheck
Self Assignment	Assignment of a variable to itself. (e.g., x=x).	CppCheck
Mutual Exclusion OR	OR operator between two mutually exclusive conditions (thus always evaluating to true).	CppCheck
Switch Redundant Assignment	Redundant assignment in a switch statement: for example, assigning a value to a variable in a case block without a following break instruction, then re-assigning another value to the same variable in the subsequent case block.	CppCheck
Dead Local Store	A statement assigning a value to a local variable, which is not read or used in any subsequent instruction.	FindBugs
Dead Local Store Return	A return statement assigning a value to a local variable, which is not read or used in any subsequent instruction. (i.e. return(x=1);)	FindBugs
Repeated Conditionals	A condition evaluated twice (e.g., x==0 —— x==0).	FindBugs
Non Short Circuit	Code using non-short-circuit logic boolean operators (e.g., & or ‖) rather than short-circuit logic ones (&& or ‖‖). Non-short-circuit logic causes both sides of the expression to be evaluated even when the result can be inferred from knowing the left-hand side.	FindBugs
Useless Control Flow	Control flow constructs, which do not modify the flow of the program, regardless of whether or not the branch is taken (e.g., an if statement with an empty body).	FindBugs

Table 4.24: Potential Energy Code Smells selected for validation.

Subsequently, we wrote for each of the patterns a pair of C++ functions, one containing a potential smell and another one refactored without that smell. For example, the “Non-Short Circuit Logic” pattern has the following two functions:

```
void NonShortCircuit_With()
{
        int count = 0;
        int total = 345;
        if ( count > 0 & total / count > 80 )
        count=0;
}
void NonShortCircuit_Without()
{
        int count = 0;
        int total = 345;
        if ( count > 0 && total / count > 80 )
        count=0;
}
```

The function NonShortCircuit With() is the one with the potential smell "*Non-short circuit logic*". The smell is in the line $if(count > 0 \& total/count > 80)$ because the AND operator is single & and so both predicates in the expressions will be evaluated at run-time. In the function, NonShortCircuit Without() the code is refactored replacing & with &&. All functions are available online for the sake of replication[13].

4.4.3 Experiment setup

Context: the WASP

The device used for the experiment is the Waspmote V1.1 (Libelium Comunicaciones Distribuidas S.L. Esso). The hardware architecture is based on a ATmega 1281 microcontroller with a CPU frequency of 8 MHz and 8KB of SRAM. It has no operating system: programs are directly loaded on a FLASH memory of 128 K. This architecture well suits our experiment because no other threads run in parallel with the chosen program, thus eliminating any software noise for the energy measurement. The device is basically a motherboard with connectors to plug in other elements such as sensors, wireless modules (ZigBee, XBee, Bluetooth), GSM/GPRS modules and a GPS (Global Positioning System) module. For this reason it is used in different fields, such as Smart Metering, Building Automation, Agriculture etc. It runs on a lithium battery (3.7V and 1150mAh), so the energy consumption of software has a key role here. To compile and load the C++ programs it is sufficient to use the IDE provided by the manufacturer and connect it to a computer via USB cable.

[13] `http:softeng.polito.it/greensmells/`

Experiment setup

The objective of the experiment is to measure power consumption and execution time on each function pair, in order to evaluate if the potential smell affects the two dependent variables. We divide the experiment in two parts: one for measuring power consumption, and another one for the execution time. Measuring power consumption and execution time for a single function is a challenging task because usually execution is too fast to get reliable data. We control this threat repeating each function 1 million times, which makes one sample. We collect 50 samples in order to reach statistical significance. Each function pair is loaded on the Waspmote and evaluated two times: the first one for the execution time, the latter one for the power consumption. No specific instrumentation was needed to obtain the execution time, because the Waspmote embeds a Real-Time Clock (RTC) with a millisecond accuracy. We measure the execution time of every loop (i.e. 50 measurements). On the other side, analysing power consumption is more complicated. The only way to obtain a precise measure of the power consumption is using a power meter. The RTC is powered by an auxiliary battery, which makes it completely independent from the main power supply. Therefore it is possible to power the Waspmote with a constant voltage ($V_G = 3.7$ V) by means of a generator, and use a shunt resistor to measure the current intensity. An analog to digital converter (ADC) connected to the PC reads the voltage drop across a resistor R of 1 Ω. The current flowing in the circuit can be computed by measuring the voltage drop on the resistor ($I = V_{ADC}/R$). The instant power consumption value can be computed as:
$P = V_L \cdot I = (V_G - V_{ADC})\frac{V_{ADC}}{R} = \frac{V_G V_{ADC} - V_{ADC}^2}{R}$
Figure 4.17 represents the circuit described. The device used to measure the power

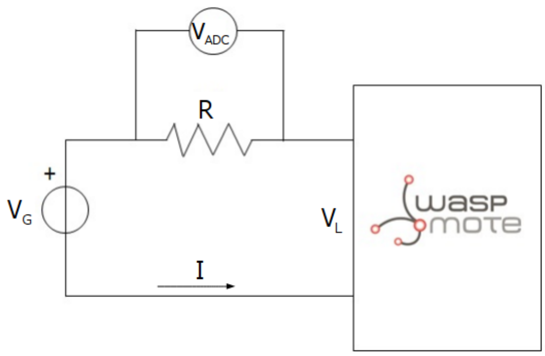

Figure 4.17. Circuit built to measure the power consumption.

consumption has a frequency of 49KHz, i.e. it gets 49000 measurements each second. In order to precisely measure the power consumption relative to the execution of the function pairs, we inserted a sleep interval at the beginning of the data acquisition to exclude the peak of device power on, and we filtered out, through a threshold, all the measurements corresponding to the idle consumption between the iterations of the function execution. As shown in Figure 4.18, the threshold filters out the transient and includes only the peaks corresponding to the actual execution of the function.

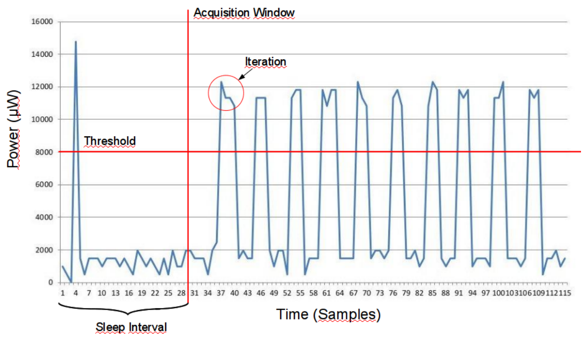

Figure 4.18. Sampling current intensity: an example.

4.4.4 Analysis methodology

For each research question we derived a pair of null and alternative hypotheses to test.

RQ13:

$$H1_0 : P_{with} \leqslant P_{without}$$
$$H1_a : P_{with} > P_{without}$$

where P is the power consumption of the function, with and without the potential smell. If the refactored version of the function consumes less than the function with the smell, the null hypothesis is rejected in favour of the alternative one. As a consequence we consider the pattern a Energy Code Smell. The hypothesis is tested with the Mann-Whitney test, given $\alpha = 0.05$.

RQ14:

$$H2_0 : T_{with} \leqslant T_{without}$$
$$H2_a : T_{with} > T_{without}$$

where T is the execution time of the two functions. If the smell has a negative impact on performance, the refactored function will be faster and the null hypothesis is rejected. In that case, we consider the pattern a Performance Smell. In order to answer RQ14, we compare which Energy Code Smells are also Performance Smells. We also use Mann-Whitney and $\alpha = 0.05$ to test the hypotheses. At the end of the experiment each function has 50 measurements of execution time and about 25 millions of power measures. Then, after filtering out values below the idle threshold (8mW), we obtained about 8 million values for power measurement, on which we ran the analysis.

4.4.5 Results

We report results on the power consumption and execution time respectively in TABLE 4.25 and 4.26. The two tables report the name of the smell, the means and their difference for both the dependent variables, the p-value of the Mann Whitney test and the difference in percentage of the power consumption (or execution time) between the execution of the code with the smell and the execution with the refactored code. We observe from Table 4.25 that all power consumptions ranged from

Smell name	**Mean with smell** (μW)	**Mean w/o smell** (μW)	**Diff. Means** (μW)	**P-value**	**Impact%**
DeadLocalStoreReturn	41241	41278	-37	1	-0.09
DeadLocalStore	40249	40205	44	$<$ **0.01**	0.11
MutualExclusionOR	40758	40772	-14	1	-0.03
NonShortCircuit	41113	41043	70	$<$ **0.01**	0.17
ParameterByValue	40967	40723	244	**0**	0.60
RepeatedConditionals	41155	41126	29	$<$ **0.01**	0.07
SelfAssignment	40952	40879	73	$<$ **0.01**	0.18
SwitchRedundantAssignment	40724	40756	-32	1	-0.08
UselessControlFlow	41051	41142	-91	1	-0.22

Table 4.25: Results of power consumption.

40mW to about 42mW. Five code patterns over nine have a p-value < 0.05 (in bold) and therefore the null hypothesis is rejected for them. The code patterns are:

- DeadLocalStore
- NonShortCircuit
- ParameterByValue
- RepeatedConditionals

- SelfAssignment

Overall the saved power consumption is less than 1%. The answer to RQ13 is: *five code patterns (DeadLocalStore, NonShortCircuit, ParameterByValue, RepeatedConditionals, SelfAssignment) are Energy Code Smells, and their impact is in the order of μW.* Focusing on performance, from Table 4.26 it becomes evident that

Smell name	Mean with smell (ms)	Mean w/o smell (ms)	Diff. Means (ms)	P-value	Impact%
DeadLocalStoreReturn	3288.76	3288.74	0.02	0.41	6.08e-04
DeadLocalStore	17707.34	17707.38	-0.04	0.66	-2.26e-04
MutualExclusionOR	3540.76	3540.60	0.16	**0.04**	4.52e-03
NonShortCircuit	3288.74	3288.80	-0.06	0.76	-1.82e-03
ParameterByValue	3288.76	3288.74	0.02	0.41	6.08e-04
RepeatedConditionals	3288.80	3288.74	0.06	0.24	1.82e-03
SelfAssignment	3288.66	3288.78	-0.12	0.90	-3.64e-03
SwitchRedundantAssignment	3540.58	3540.62	-0.04	0.65	-1.13e-03
UselessControlFlow	3288.80	3288.74	0.06	0.24	1.82e-03

Table 4.26: Results of execution time.

there is no difference in execution time. The null hypothesis is rejected only for MutualExclusionOr, however the magnitude order is μ seconds. We also notice that DeadLocalStores are about 5 times slower. Thus, our answer to RQ14 is: *Energy Code Smells are not Performance Smells.*

4.4.6 Discussion

We identified five smells which provoked a higher power consumption of the Waspmote in the use cases prepared for the experimentation. However, we observe that the saved power is less than 1 %. A first motivation resides in the implementation choices: the function pairs executed only differ in a single instruction, and the operations are done with primitive types (e.g., integer). The motivation of such implementation was the exclusion of any possible confounding factor in the analysis, but the drawback of such a choice is a very small achievement in energy efficiency improvements. Let us take dead stores as an example: the smell *DeadLocalStore* is implemented with an integer (we save a value on a variable and immediately overwrite it with another integer). Using a struct with several members is totally different and might lead to a higher impact, because the resulting compiled code requires the CPU to produce more instructions and interact more intensively with the memory. If increasing the complexity of the data structure will result in still negligible power consumption saving, the next step is to increase the logical complexity of the function, i.e. comparing complete algorithms that are functionally equivalent but differ in the implementation. A further step is to move the focus towards the comparison of functionally equivalent design choices. Understanding the impact of Energy Code Smells over real power consumption could also contribute to build more precise models of the power consumption of software. As a matter of fact, it may be possible to categorise software instructions beforehand in terms of energy efficiency, then subsequently use this information in order to predict the resulting energy efficiency of a complete software product. Yet another research direction that is suggested by this first leap is: can the impact of Energy Code Smells be higher in code that drives a hardware resource with higher energy needs? For instance the impact on the code that handles the GPS transmitter is expected to be very different from the one used in this experiment, where the small functions use only CPU and RAM, besides not using them in an intensive way. The same investigation approaches can be applied to the domain of execution time. As can be noticed from the results, all the execution times are equal, with an exception given for the *DeadLocalStore* function pairs. We have observed that Energy Code Smells do not degrade the performances, but we cannot generalise the findings for more complex code structures and usage scenarios, with different hardware resources involved (e.g, sensors).

Threats to validity

In this section, we expose the threats to validity that might affect our study. As regards construct validity, our main threat regards instrumentation. We carefully evaluated the precision of our measures, comparing them with the specifications from Waspmote manufacturers. During our experimentation, the difference between actual and expected values was negligible and inside the specified ranges. Internal validity is represented by confounding factors such as other processes running during execution. However, the Waspmote does not have an operating system and the only thread in execution during the tests is the code loaded. As regards external validity, we do not aim at generalising our results to a family of embedded devices. This study aims at assessing the existence of the Energy Code Smells in a single context: other empirical validations are necessary for other environments or devices.

4.4.7 Considerations

This is an exploratory study: we defined for the first time the concept of Energy Code Smells and we performed a first validation to understand not only the impact, but also the boundaries of the concept. We identified some Energy Code Smells starting from code patterns implemented by two common Automatic Static Analysis tools - namely, CppCheck and FindBugs. We performed an experiment, on an embedded system, in order to assess the energetic impact of those code patterns, also taking into account the impact on execution time, to determine whether Energy Code Smells are also performance smells. Our experimental results showed that some of the code patterns actually have an impact over power consumption. This impact, however, is in the magnitude order of μW. Our future research works will be devoted to analysing more complex data structures and using hardware resources, which could increase this impact with respect to the overall power consumption. As regards time analysis, only one pattern had an actual impact over execution time (a few μ seconds), and it is not identified as an Energy Code Smell. Thus, we conclude that Energy Code Smells are not Performance Smells. However, results suggest that the target and applicability of Energy Code Smells should be refined with further investigations. The lessons learned in this exploratory study let us identify several research threads that the research community might address, such as the identification of Energy Code Smells that are higher-level constructs and use more complex data structures, the identification of Green Design Smells and the use of more complex systems as test beds. Finally, the experimental results that will be collected might be also be used to build more precise models of the power consumption of software.

4.5 Self- adaptation

The increasing proliferation of mobile devices and the intention of defining universal standards related to the mobile market, motivate many companies to implement context-aware standards, with the purpose of stimulating a rapid and wide adoption of a variety of useful applications. A context-aware system has to be able to combine contextual information, which is related to the bounded environment. This info, called context data, is any information that can be used to characterise a specific entity situation [4].In this way it is possible to describe the actual situation, by determining some automatic behavioural variations or by notifying the user about some specific event. This kind of system has to be constantly in execution to gather raw data and to execute different types of operations based on context reasoning. Context-Aware services collect contextual information automatically. With a wide range of possible user situations, it is important that services have a way to adapt appropriately to best support low battery scenarios. A system is context-aware whether it uses the context to provide relevant information and/or services to the user, where relevancy depends on the user's task. Many approaches are not completely dynamic, flexible or effective when we need to automatically match battery consumption requirements in differing contexts. A Context and Energy-Aware system has to be flexible and able to react to environment variations. Many features of modern devices like high processor speed, more efficient displays, more powerful data storage and WiFi/GPRS/UMTS network adapters, and specific hardware to enable advanced 3D graphics, considerably influence the device energy costs. Current approaches are not able to implement energy-aware self-adaptation because they do not consider aspects related to context management policies. Context can be used to find a lower energy consumption performance as well as implementing proper adaptation policies. An energy-aware context broker has to deal with:

- Sensing: to detect as much contextual information as possible,
- Transmission: to send gathered information to a context platform (for further processing),
- Adaptation: to manage the energy cost caused by *sensing* and *transmission* phases.

The context broker should be able to collect (and to send) context data only when really needed to avoid useless duplicated data management. It is also fundamental to include the battery charge state in the optimization strategy the context broker is adopting: the context broker shall be able to manage the information granularity depending on the battery level. It is also indispensable to respect the following fundamental principles [53] in order to reduce the energy consumption to the lowest possible level:

- less work requires less energy,
- event programming avoids polling,
- multi-core environment programming,
- periodic timer should be avoided and
- the system should be scalable.

A context-aware application has to be able to minimize its operations in situations where the device is running out of energy [65]. In order to cut the device runtime operations, it is important to:

- to implement an efficient algorithm or to modify an existing one to reduce operations have to be carried out to a minimum,
- to improve the compilation process by introducing optimizations based on target processor and

- to prefer a compiled and optimized code instead of an interpreted one.

Nowadays, operating systems for mobile devices provide different APIs to check the battery state (e.g. battery charge percentage, battery technology and temperature). Software components can be notified whenever batteries state changes. Energy Awareness is concerned with several aspects: the quantity of energy that is used, what this energy is used for, where it comes from, the resulting effects (e.g. environmental impact, resources consumption). In addition it poses the problem of how to reduce the energy consumption and its collateral effects. Even if software does not consume energy directly, it does, however, have a direct influence on the energy consumption of the hardware underneath. As a matter of fact, applications and operating systems indicate how the information is processed and, consequently, drive the hardware behaviour. We think that writing energy-efficient code in a mobile environment can be more appreciated by users because there is a direct effect on their mobile batteries lifetime. The Energy-aware middleware we are going to introduce in this paper uses certain resources (i.e. GPS, Bluetooth) related to different operations that have a high impact on energy consumption. *gLCB* implements features, which focus on reducing the energy consumption by avoiding the execution of redundant operations [12]. *gLCB* is responsible for retrieving context information through device hardware sensors, and it is also responsible for delivering such data to a Context Awareness Platform (CAP) [30]. The CAP allows the collection of context data from users' devices. Context data can eventually be processed by other components of the platform and become high-level context information (e.g. GPS coordinates can be translated into a civil address). This process is called reasoning [13]. Figure 4.19 describes the functional architecture of our CAP. From the figure we can deduce how context is transformed from raw data into high-level information and how high-level information is exposed to external applications.

Context Capturing Layer

In this layer, raw context data are retrieved from the available device sensors (e.g. GPS, phone, Bluetooth, WiFi, etc) and aggregated. The aggregated data are then asynchronously transferred to the CAP. This layer of abstraction collects context data in a fast and economic way, in order to integrate heterogeneous information sources and consequently support various protocols and different kind of data formats. This is the layer where *gLCB* acts.

4.5.1 Context Analysis Layer

gLCB captures the low level data representations, which may not be meaningful to applications, and sends them to the Context Broker. The CAP will modify

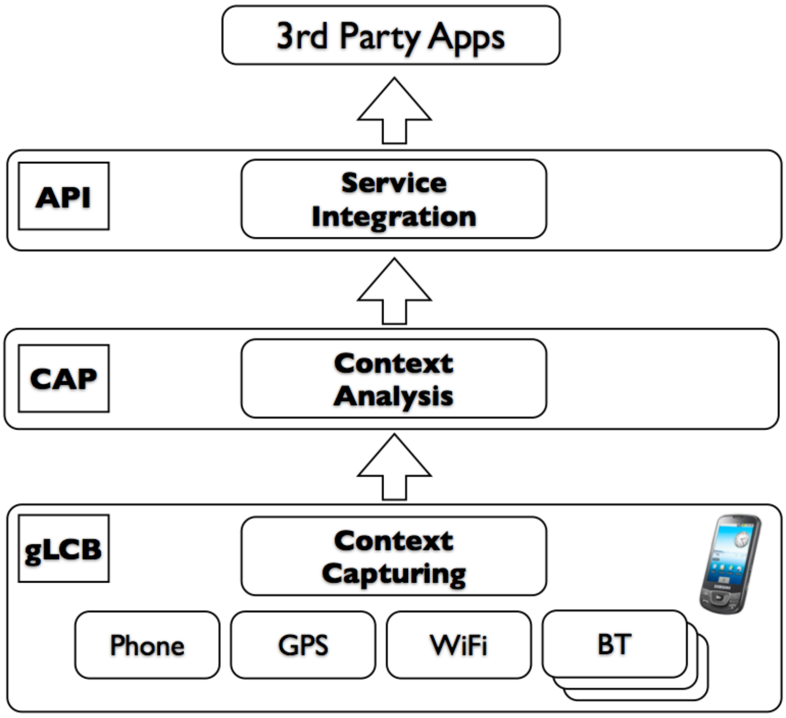

Figure 4.19. Context Awareness Platform: from raw data to high-level information

this information in high-level representations which are easier to interpret and to use (e.g., an address is more significant that GPS coordinates). In context-aware architectures, the reasoning component elaborates raw data and generates high-level information. The reasoning process may require significant computational effort so it is usually a server-side operation. Besides reasoning, the CAP may also try to learn user behaviours: this technique is known as learning. Learning usually involves the analysis of a context history database and may be executed offline or online. Offline learning is a batch process, which runs periodically (daily, weekly, etc.) and is applied to the whole context history. Online learning is executed at every context change and receives continuous feedback.

Service Integration Layer

This layer exposes context information towards the service platforms via API. The interfaces exposed by the CAP also allow third party applications to provide context data. As pictured in figure 4.20 the actors involved in the CAP are:

- Context source: a component, which feeds context data into the CAP. A source typically provides raw data (e.g. a mobile phone);

- Context consumer: a component which uses context information (e.g. an application);
- Context provider: a context source which is also capable of producing higher level information by acting as a consumer of raw data and a source of high level information;
- Context broker: the component which stores context data.

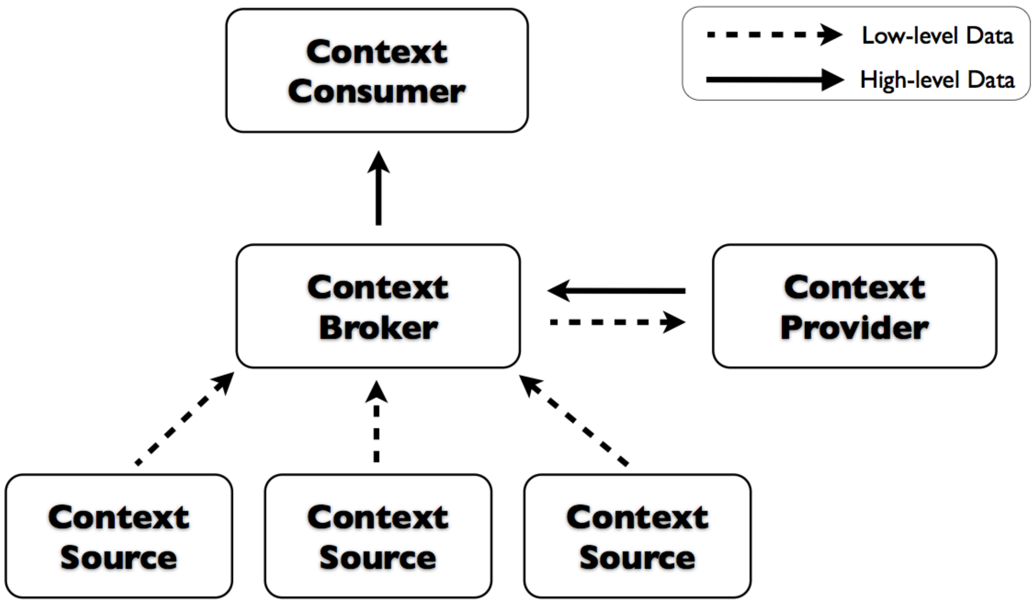

Figure 4.20. Context Awareness Platform: actors

Context Capturing Layer

More than one context consumer and context source can access the platform to request and send context data. Specifically the context data are sent from applications executed in mobile devices. In this way, it is possible to perform data mining and clustering operations. Context Consumers can retrieve context through synchronous requests to the Context Broker or asynchronously by registering to context changes. This second scenario optimises the allocation of resources and the network traffic: applications do not need timers to poll the CAP, the CAP triggers applications only whenever a meaningful context change takes place.

4.5.2 gLCB

The middleware software we developed for Android OS, is based on two levels: the first one is associated to the local context broker, and the second one to the sensors layer. The lower level establishes a communication channel to the higher level by sharing an interface. Mobile context-aware applications can rely on the local context broker to retrieve context data. To do so, the application is requested to bind to the LCB [60] service; when the binding is established, the application can perform context queries which will result in the retrieval of context data from the current device (if the requested information is available locally) or from the CAP. The local context broker manages its sensors via a Sensor Manager (Figure 2). The Sensor Manager is responsible for:

- discovering new sensors,
- the management of sensors life cycle and,
- the management of their settings.

This component also exposes an interface, which allows the local context broker to perform one-shot requests to sensors or to subscribe to context data variations. Every sensor is installed as a service and runs in the background on the device.

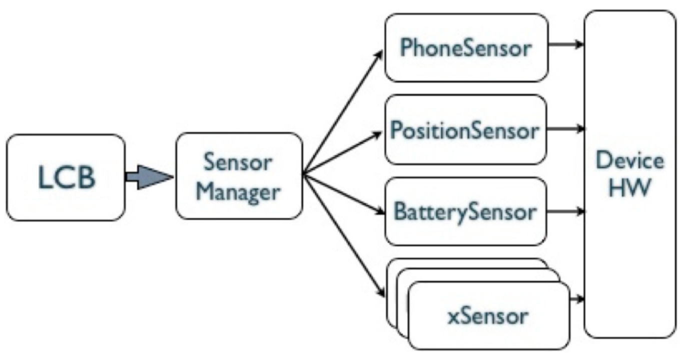

Figure 4.21. LCB sensors management

Critical aspects

We needed to guarantee a starting sensors mechanism in order to provide the independence of the components involved during the application execution. Independence is crucial as the number of developed sensors may vary and the test phase cannot depend on a newer release of *gLCB*. In this way the Android operating system is able to recognize each sensor as an independent application. The other critical aspect concerns the way in which data are obtained from sensors. A possible solution should be the sequential scanning of every sensor and the consequent data publishing in the server side. This approach introduces some critical problems mainly related to:

- High device battery consumption;
- Static data search and data publishing based on specific movement;
- Redundant context data transmission;
- A single monolithic application including every sensor.

Limited Search and Publishing

Context data search and publishing are the most relevant stages related to device battery consumption. Sensors that manage data provided by the operating system (e.g. IMEI number, ringtone volume etc.) do not reveal critical problems related to information management because such information is fixed. On the other hand, the scanning of new wireless networks, the nearby Bluetooth devices or the geographic position calculation, represents expensive operations in terms of energy consumption. These operations have to be limited whenever unnecessary.

Asynchronous Sensors Association

We created a sensor starting mechanism that was not dependent on the main application starting mechanism. In this way, new sensors are independent services associated with different starting components. Each sensor is characterized by a component called BroadcastReceiver. This component is an independent class that intercepts a specific system message called Intent. *gLCB* broadcasts an Intent whenever it is launched. Every sensor is able to intercept such intent and to react with a "Sensor Available" intent which notifies *gLCB* about its availability. In this way, it is possible to associate an undefined number of sensors with the main application. The only restriction is that they must be installed in advance on the device. Table 4.27 lists every available sensor.

Sensor	Description
WiFi	List of WiFi networks
DeviceActivity	Information about current applications running
Location	Geographical user position
DeviceStatus	Terminal publishing status
Phone	GSM or UMTS cell on which is connected
Bluetooth	Bluetooth neighbours
Data	Connectivity device info
Call	Call status

Table 4.27: Sensors Description

Asynchronous Context Data Publishing

Another critical issue about power consumption is the redundant publication of unchanged context data. A periodical sequential scanning of all sensors will cause a waste of energy as every sensor is activated at each scan. A possible solution to this problem is to implement a more efficient context scanning algorithm, e.g. an event-based algorithm which activates the sensors only in response to specific events. Table 4.28 lists every available sensor and the events that trigger the data update on the server side.

Sensor	Events triggering a data update
WiFi	Sensor Start
	Out-of-date Context Data
	New WiFi Networks
DeviceActivity	Sensor Start
	Out-of-date Context Data
Location	Sensor Start
	Out-of-date Context Data
	Movement Greater Than Threshold
DeviceStatus	Sensor Start
	Out-of-date Context Data
	User Profile Change
Phone	Sensor Start
	Out-of-date context data
	New Cell Connection
Bluetooth	Sensor Start
	Out-of-date Context Data
	New Bluetooth Handsets
Data	Sensor Start
	Out-of-date Context Data
	Connectivity change (3G or WiFi)
Call	Sensor Start
	Out-of-date Context Data
	10 Calls (Incoming or Outgoing)

Table 4.28: Sensors Events Description

All the sensors share two events: *Sensor Start* and *Out-of-date Context Data*. The first event means that the sensor collects and sends context data whenever the sensor is started. The second event means that after a certain period the sensor must refresh the context data because it is deleted by the CAP. This time period can be managed to increase or decrease the number of context updates regardless of data variation. To save energy we introduced a new parameter called *mindelta*. With this parameter the user can set, for each sensor, the minimum time period which must elapse between two updates. Mindelta must be lower than the data expire time to ensure the context data availability. Basically, by managing *expire time* and *mindelta* parameters, the user can find a good trade-off between data processing and energy efficiency. In this way, the data stored on the server side is stable, updated, and available for Context Consumers.

Different Update Policies

We created seven user profiles: *VERY LOW* (Table 4.29), *LOW* (Table 4.30), *NORMAL* (Table 4.31), *HIGH* (Table 4.32), *AUTO*, and *CUSTOM*. Each one can change the gLCB behaviour in terms of context data searching and publishing by managing:

- the number of active sensors;
- the expire time parameter;
- the mindelta parameter.

The *LocationSensor* is more complex than the others and manages three further parameters, which trigger a context data update:

- enables or disables the GPS module;
- computes the distance in meters between the actual value and the previous;
- computes the accuracy ratio between the actual value and the previous.

The AUTO profile selects the best profile from *VERY LOW*, *LOW*, *NORMAL*, *HIGH*, profiles basing upon the actual battery level (see Table 4.33). By selecting the *CUSTOM* profile, the user can decide the updating policy for every sensor.

Sensor	State	Texp(s)	Mindelta(s)
Phone	ON	3600	180
Location	OFF	-	-
WiFi	OFF	-	-
Bluetooth	OFF	-	-
DeviceInfo	ON	3600	3600
DeviceStatus	ON	3600	1200
DeviceSettings	ON	3600	3600
DeviceActivity	OFF	-	-
DataSensor	ON	3600	1200

Table 4.29: VERY LOW user profile configuration

Sensor	State	Texp(s)	Mindelta(s)
Phone	ON	3000	180
Location	ON	3000	120
	D=5000m		
	A=10		
	GPS OFF		
WiFi	OFF	-	-
Bluetooth	OFF	-	-
DeviceInfo	ON	3600	3000
DeviceStatus	ON	3000	180
DeviceSettings	ON	3000	360
DeviceActivity	OFF	-	-
Data	ON	3600	180

Table 4.30: LOW user profile configuration

Sensor	State	Texp(s)	Mindelta(s)
Phone	ON	1200	120
Location	ON	1200	120
	D=5000m		
	A=2		
	GPS ON		
WiFi	ON	1200	120
Bluetooth	ON	1200	300
DeviceInfo	ON	3600	300
DeviceStatus	ON	1200	120
DeviceSettins	ON	1200	120
DeviceActivity	ON	1200	120
DataSensor	ON	3600	120

Table 4.31: NORMAL user profile configuration

Sensor	State	Texp(s)	Mindelta(s)
Phone	ON	900	30
Location	ON	900	0
	D=250m		
	A=1		
	GPS ON		
WiFi	ON	900	120
Bluetooth	ON	900	120
DeviceInfo	ON	3600	120
DeviceStatus	ON	900	60
DeviceSettings	ON	900	60
DeviceActivity	ON	900	60
DataSensor	ON	3600	30

Table 4.32: HIGH user profile configuration

Interval	**Profile selected**
Re-charger connected	HIGH
100% - 61%	HIGH
60% - 41%	NORMAL
40% - 16%	LOW
15% - 5%	VERY LOW
<5%	LCB Switched off

Table 4.33: AUTO profile behaviour

Application Execution

When started, *gLCB* loads the configurations used during the last execution. At the first start *gLCB* triggers the *HIGH* profile. For every sensor, it is possible to identify the update state, name, the time of the latest update attempt, and the outcome result. There are three possible outcomes:

- Update OK,
- KO.net (no publishing because of net error) and
- KO.cb (no publishing because of context broker error).

After that, *gLCB* displays the actual selected profile and if it is selected manually or automatically, shows the number of successful publications, the date and time in which the application has been started. In figure 4.22, we can identify the user has

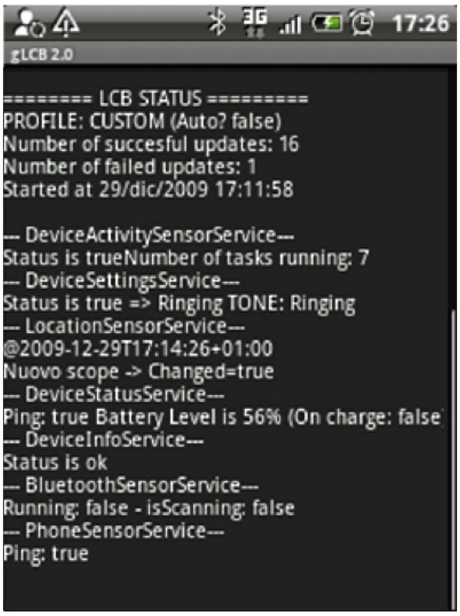

Figure 4.22. A screenshot of gLCB log activity

7 tasks in execution, the ringtone is enabled, new geographic position information was revealed, the battery level is at 56%, *DeviceInfo* and *PhoneSensor* sensors are enabled, and *BluetoothSensor* sensor is not enabled. When the user switches off *gLCB*, a special context publication will be carried out to the server, which recognizes *gLCB* is switching itself off and cancels every context data related to that device. This avoids other Context Consumers reading data that is not valid.

4.5.3 Validation Criteria

We chose an empirical approach to demonstrate the algorithm optimisation effectiveness and we employed two different techniques to measure our middleware energy consumption. The first technique – "user side" – measures the time required by each profile to run out the phone battery (Time Measurements), while the second –"lab side" – aims at measuring the instant power consumption of each different user profile (Instant Power Measurements). For both the approaches we scheduled the execution of a set of automated tests which did not require the user participation in order to get a better measurement precision. The reference device is a Smartphone

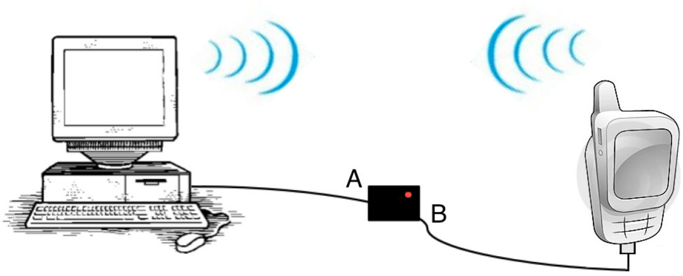

Figure 4.23. BatterySwitch

Samsung Galaxy (i7500) connected to the net through Telecom Italia Mobile (TIM) 3G network.

4.5.4 Results

Time Measurements

The procedure we adopted to perform the battery duration measurement consists of the following steps:

- charge the battery until maximum battery level;
- select a specific *gLCB* user profile and let it start collecting data;
- record the time instants when the battery charge level changes;
- stop when the battery charge level reaches a predefined minimum value (5%);

Such a procedure lends itself to an easy automatic repetition of several measurement cycles. We carried out measures for each profile and each configuration automatically for thirty times in order to have statistical evidence. In addition to the total discharge time, the above procedure, although in a quantitative way, can show how the battery behaviour changes. From the user perspective it is a very useful finding. In order to conduct the above measurement procedure we built a dynamic battery charger, which allows starting or interrupting the battery charging through a specific command sent by a controlling PC. We called this particular battery charger *BatterySwitch* and it has two USB ports as shown in figure 4.23:

- One port is connected to a pc (USB PORT A),
- The other port is connected to the mobile phone to manage the charge of the battery (USB PORT B).

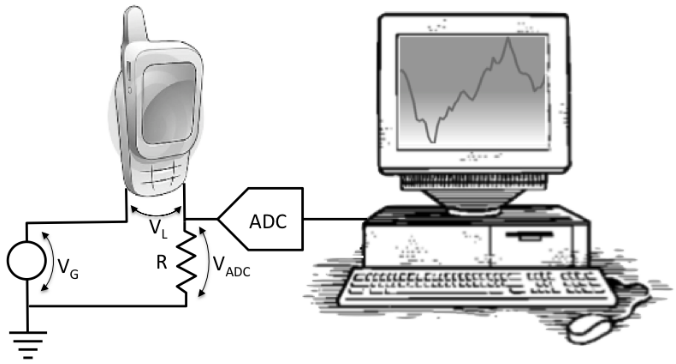

Figure 4.24. Circuit developed to get instant power consumption values

BatterySwitch has a small firmware that can manage the supply of the USB port B, which will be connected to the mobile phone. *BatterySwitch* is detected as a HID-interface once plugged into a pc and it is possible to interact with it through a simple application. For this reason it is needed a program that will:

- Be executed on the pc,
- Analyze the USB stack,
- Detect the device,
- Send bit '0' to start charging the battery,
- Send bit '1' to stop charging the battery.

This simple program has a thread listening to the port number 20248. The smartphone opens a socket to that port and it sends character 'a' or character 's'. The program will send respectively bit '0' or bit '1' to *BatterySwitch* to start or stop charging the device battery. It is possible to set up a group of configurations so that, at the end of each measurement, the device publishes the collected data to the server, and starts the next measure.

Instant Power Measurements

To measure the instant power consumption, we followed the procedure below:

- select a specific *gLCB* user profile and let it start collecting data;
- record the power consumption;
- stop after a predefined time (1 hour) has elapsed .

The device we used to measure power consumption is presented in figure 4.24. A power supply behaves like an ideal voltage generator with a constant 5 V tension. An analog to digital converter (ADC) connected to the PC reads the voltage drop across a resistor $R = 1\Omega$. The current flowing in the circuit can be computed by measuring the voltage drop on the resistor ($I = V_{\mathrm{ADC}}/R$). The instant power consumption value can be computed as:

$$P = V_L \cdot I = (V_G - V_{\mathrm{ADC}})\frac{V_{\mathrm{ADC}}}{R} = \frac{V_G V_{\mathrm{ADC}} - V_{\mathrm{ADC}}^2}{R}$$

The ADC sampling frequency is 49 MHz and each user profile is measured for 1 hour. Each measurement is repeated 30 times to get statistical evidence. To ease the collection of all these sets of measures execution, we developed a simple scheduler on the mobile side, which runs *gLCB* with different user profiles.

Time measurements

The complete battery discharge curves for each profile are presented in Figure 4.26. We can easily appreciate the non-linearity of the behaviour, which represents the main reason that led us to consider the total discharge time. In the figure, the rightmost abscissa reached by each profile represents the time required to achieve a 5% charge level: this is the time we consider as the total discharge time. Table 4.34 reports, for each user profile, the total discharge time. The different profiles essentially differ for the frequency of context data updates transmitted by the gLCB to the CAP server, the second column presents the average number of updates per hour. The average duration of our Samsung Galaxy battery in stand-by (without applications running) is 15 hours 33 minutes and 22 seconds. Since this value is expected to decrease according to the selected user profile, table 4.34 also reports the discharge variation in percentage with respect to the stand-by configuration (last column). We can observe that as the number of publications increase, the mobile phone battery discharge time decreases linearly. The battery discharge time (t) is linked to the update frequency (f) by a linear relationship:

$$t = 13h : 28m : 14s - 39m : 33s \cdot f$$

The above equation captures the observed behaviour precisely ($R^2 = 97\%$). The offset of the equation does not correspond to the stand-by time since the execution of gLCB, even without updates, consumes power. Also the profile labelled "AUTO" does not match the linear relationship because it dynamically adapts the update frequency to the battery charge level. Actually, with an average update frequency of 7.54 we could have expected a discharge time of 8 hours and 30 minutes, which is shorter than the one achieved using the AUTO heuristic. We can also notice

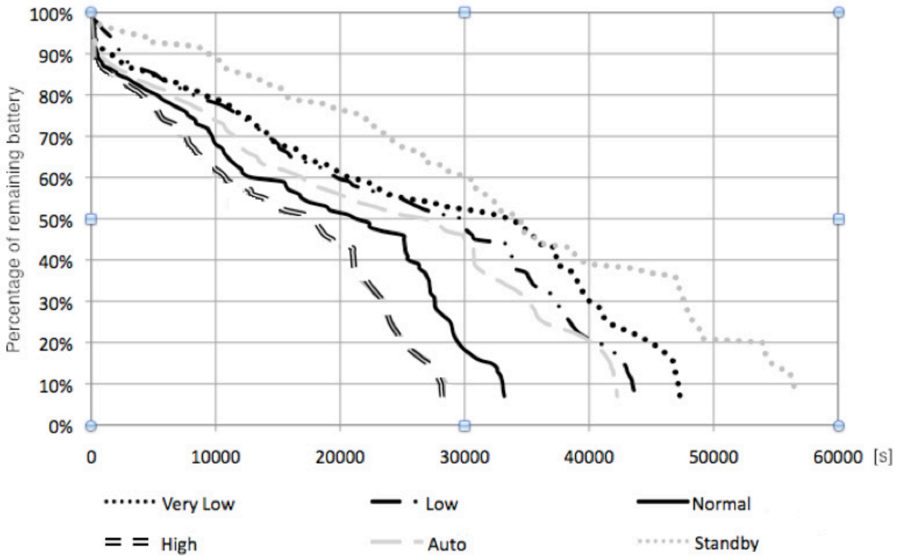

Figure 4.25. Discharge battery curves per user profile

that the *LOW* profile needs to be reconfigured because its values are too close to the *VERYLOW* profile. The energy consumption measures we carried out do not consider either battery quality and temperature, nor which resources are used by *gLCB*. This is just an approximate view, but despite that we can see our expectations have been verified: different user profiles have different energy consumption patterns.

Profile	**Updates per hour**	**Discharge Time**
STANDBY	-	15h 33min 22sec
VERYLOW	1,25	13h 8 min 8 sec
LOW	1,32	12h 7 min 35 sec
NORMAL	6,35	9h 13 min 6 sec
HIGH	8,48	7h 55 min 55 sec
AUTO	7,54	11h 41min 33sec

Table 4.34: Battery Duration and Updates per User Profile

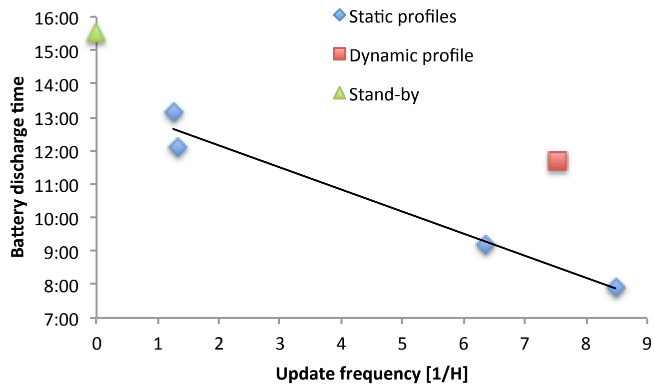

Figure 4.26. Update frequency vs. discharge time

Instant Power Measurement

The measurement with a fixed power supply allowed us to collect instant power figures during the operations of gLCB. Table 4.35 reports, for each profile, the mean instant power measured during the test time. In addition the last column shows the percentage of increment w.r.t. the stand-by profile. The $AUTO$ profile could not be measured because it adapts on the basis of the battery charge level, but in this configuration the power is provided by an external power supply and not by a battery. Figure 4.27 contains the box plots of instant power measured, it shows the actual distribution of power consumption in each experimental run. To test whether the difference among the different profiles is statistically significant also in presence of the measured variance, we performed a set of statistical tests. We selected non-parametric tests due to the non-normal distribution of the data. In particular we applied the Kruskal-Wallis test to detect overall difference among the profiles, and the Mann-Whitney test for pair-wise comparisons. According to the Kruskal-Wallis test there is evidence of a significant difference in terms of power consumption among the profiles (p-value < 0.001). In more detail, on a pair-wise comparison basis, we observed a significant difference between the following pairs: Normal and Low ($p < 0.001$), Low and Verylow ($p = 0.003$), and Verylow and Standby ($p < 0.001$). The magnitude of the difference, measured in terms of standardised effect size can be considered large. The only non-statistically significant difference is between High and Normal.

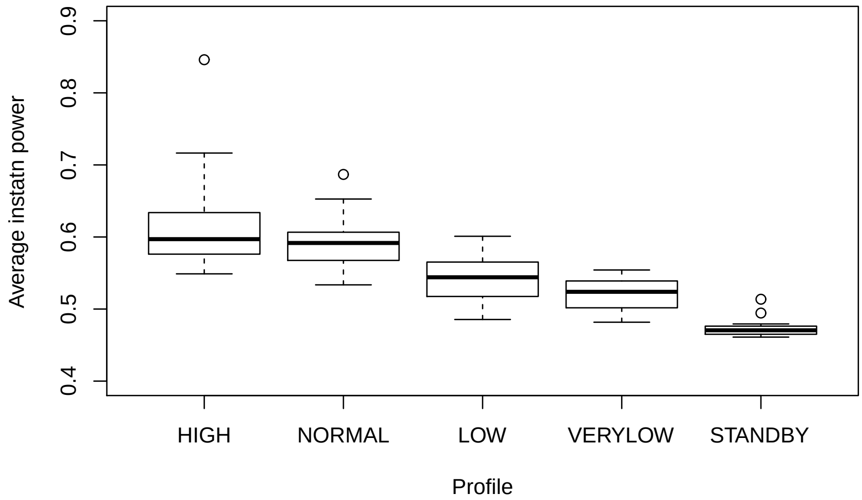

Figure 4.27. Box Plot of profiles average instant power consumption

Profile	**Power**
STANDBY	473 [mW]
VERYLOW	522 [mW]
LOW	544 [mW]
NORMAL	594 [mW]
HIGH	617 [mW]

Table 4.35: User Profiles average instant power consumption and variations

4.5.5 Considerations and future scenarios

The main goal of context-aware systems is to provide relevant information, and/or services, based on current user context. In this paper we analysed the energy consumption behaviour of *gLCB*: a context-aware middleware, which runs in background in Android OS based mobile phones, and sends context information to a remote platform. We described the architecture of *gLCB*, which is designed to adapt its behaviour on the basis of the remaining battery information. We analysed some principles based on Energy-Aware software to determine how mobile devices should behave according to scarce or plentiful energy, and how context information can be used to infer energy consumption policies. These aspects are important to improve the efficiency of context-aware systems in terms of energy consumption. The energy consumption analysis involved two different empirical experiments: in the first one we measured the average time employed to run out the mobile device battery in each user profile, while in the second one we measured the average instant power consumed by each user profile. Since the behaviour of a mobile terminal in motion is not predictable because the network signal received is not stable, we consider average values and we repeated each experiment 30 times. In this way we normalised the data and obtained statistical evidence. Considering the results obtained, we provided information related to:

- the battery average discharge-time,
- the average power consumption,
- the number of context updates per hour

of each user profile. These results have been associated with the battery duration in standby conditions, as a reference point to compare how efficient the proposed approach is. Time measurements show that AUTO profile provides the best performance between energy consumed and data uploaded. As a first result of the time measurements we are planning to reconfigure the LOW Profile because there are big differences, in terms of average battery duration, average power consumption, and context updates per hour, from VERYLOW and NORMAL profiles. Then we can see that the impact of software in power consumption is real and a self-adaptive behaviour can help to increase the battery life of a mobile device. Even if the battery discharge measure needs a very complex system, we use these results only in a qualitative way. We gathered the instant power consumption value to get a more detailed view about the device energy consumption. This deeper analysis, followed by a statistical treatment of the data collected, highlights a significant difference in terms of power consumption among the profiles. Other information such as CPU usage percentage, battery temperature, technology and voltage, could be used to evaluate how long and how many resources the application uses. It could be possible to improve even more the efficiency of *gLCB* by considering single resources usage. These results lead to defining a technique for mobile applications, which exploits self-adaptation driven by real-time energy consumption data, to increment battery life. The requirement is that energy consumption data should be collected from the Hardware Layer up to the Application Layer in real time (e.g. by built-in power meters) as shown in Figure 4.28 Once energy consumption data is available at runtime, it is possible to apply different self-adaptation techniques such as: Conditional Operation, Function Selection, Control Flow Adaptation, etc. as described by Peddersen and Parameswaran in [80]. The technique I am proposing in this paper works as follows: the Operating System Layer manages the overall energy consumption data coming from the Hardware Layer by means of a Smart Power Management module called SPM [8]. SPM divides the overall energy consumption

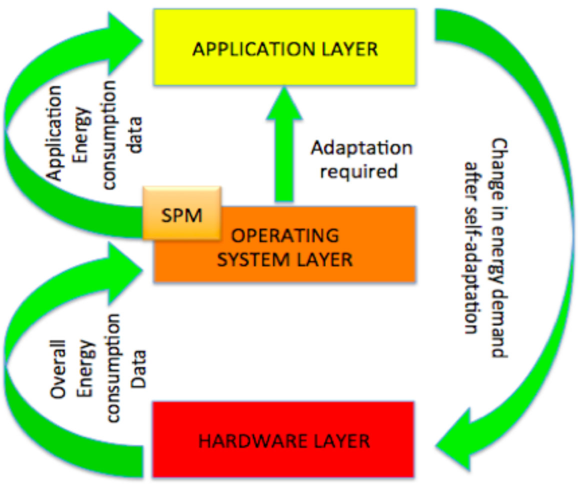

Figure 4.28. Energy Consumption Data Flow

data based on current usage of running applications, and sends this data to each application. SPM also imposes constraints and target thresholds based on different user profiles. When necessary, SPM notifies the apps exceeding the threshold. The notifications are intended as asynchronous messages sent by the Operating System (such as Android Intents) so that every application can receive them. If an application can manage the notifications sent by SPM, it will modify its behaviour exploiting self-adaptation techniques. The notified applications will finally find the best trade-off between energy consumption and features provided to the user. The data, on which the self-adaptation is performed, is the instant energy consumption of the peripherals that compose the device. This mechanism can be better explained with an example: when a self-adapting application APP1 starts, it sends an asynchronous message to SPM to subscribe to future notifications from SPM. Based on the user profile defined by the user, SPM will send the application the energy constraints in a string format through a notification as described in Figure 4.29. An example should be the following: "Resource: GPS, Max Power: 380mW, Max Time over power limit: 60sec". If APP1 crosses both the thresholds, SPM will send a warning message: "Warning, Resource: GPS, Limit: 380mW, Measured: 440mW". This warning triggers APP1 self-adaptation. If the desired result is not reached after a predefined number of iterations, or if an application has not subscribed to receive notifications, SPM will notify the user about the energy issue, and he/she can choose whether to ignore the message or kill one of the proposed applications. After the implementation phase, further studies are needed to verify the energy overhead required by this technique. The methodological approach I

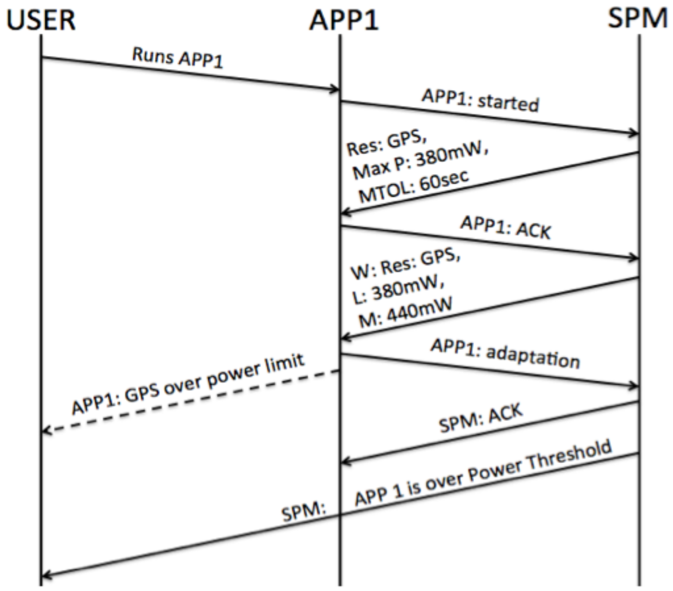

Figure 4.29. Smart Power Management operation

adopt for the validation of this technique is that of empirical software engineering [106] [91], performing experiments and case studies. Managing energy consumption data in real-time by exploiting it to drive self-adaptation in mobile applications is the main contribution of this research. This technique will enable mobile developers to be more aware of the energy efficiency of their code. Moreover it will improve the battery life of mobile phones providing a better user experience.

Chapter 5

Conclusions

This dissertation thoroughly investigated the responsibility for the software on the energy consumption of a device. The contributions can be followed as five research goals:

- **RG1. Is it possible to measure the energy consumption of an application?**
- **RG2. Could Energy Efficiency be considered as a software non-functional requirement?**
- **RG3. Is it possible to profile the energy consumption of a software application?**
- **RG4. Is there a relationship between the way a program is written and its energy consumption?**
- **RG5. Is it possible to use the energy consumption information to trigger self-adaptation?**

As a starting point, the literature has been surveyed in order to: introduce a taxonomy of concepts related to energy and IT, present recent data on energy consumption trends organised according to the taxonomy, present some guidelines to write energy-efficient software organised according to the taxonomy, underline what is missing and what should be done in future research. This work, described in Chapter 2 provides a twofold contribution represented by the taxonomy. Through this taxonomy it is possible to repeat this work in the future and compare results in terms of energy consumption and in terms of verification of the selected guidelines, which nowadays are little more than simple advice.
A framework, which is useful to researchers and practitioners, who intend to create

energy-efficient software has been defined in Chapter 3, and it uses software profiling tools (described in Section 3.2) in order to measure the power consumption of the device when running the code under test. The framework provides two kinds of strategies (not mutually exclusive) to reduce the energy consumption: the first one is related to the source code refactoring and the progressive identification and removal of the energy code smells, while the second one provides design changes in order to introduce the user profiles and the self-adaptation. Once the changes have been made, the developer can again use software profiling tools in order to verify the effective energy consumption reduction.
Given the growing interest with regard to the energy consumption of software applications it is good to wonder if energy efficiency can be considered as a non-functional requirement. Chapter 3 addresses the problem and proposes an extension of SQALE in order to introduce energy efficiency. The suggestion is to introduce Energy Efficiency as a sub-characteristic related to the main characteristic "Efficiency". Energy efficiency cannot be a characteristic itself, because it is not an activity in the typical software lifecycle, but it is a sub-characteristic of the second type.
It was also necessary to assess whether it was possible to empirically prove a relationship between the software and power consumption. These experiments described in Chapter 4 have involved a small datacenter, desktop PCs and mobile devices. The result is that there is a relationship between software and power consumption but the main issue that affects this kind of experiment is the lack of generalisability of results, which depend strictly on the tested device. However, the various scenarios described in Chapter 4 demonstrate that energy consumption varies depending on the software running, and this allowed us to continue our studies on the topic. After this study it is necessary to empirically prove that the way in which a program is written can alter the power consumption of the device. The goal of the study described in Section 4.4 is how to optimise software by identifying code patterns that use the hardware resources in a sub-optimal way. These code patterns ought to be refactored in order to improve the energy efficiency of the software at run time.
The progress on this topic led us to wonder if it was possible to design a software, which can automatically adapt to the energy situation in which the device is in that moment. To do this it is possible to use the energy information as context information and to create user profiles, which modify the behaviour of the application according to the energy information gathered from the device. This approach, called self-adaptation, is explained in Section 4.5. This Section also shows some empirical experiments that demonstrate the effectiveness of this approach.
The efficiency of a software application can be highlighted by applying energy labels to it. Nowadays, we have energy labels for almost everything: home appliances, ICT devices, buildings, etc. A typical use of energy labels is related to buildings and home appliances, which are ranked by their energy consumption. In the first case, building owners or facility managers can see where energy is wasted and eventually

they can take targeted countermeasures in case of excessive energy consumption. In the second case, a user can decide whether to buy a home appliance or not, and they can select the most energy-efficient, even if it has a more expensive price tag. These mechanisms make the user aware of the energy issue and in the long run they change the users behaviour. There are many eco-labels related to IT equipment (such as Energy Star), however these do not cover the detailed device usage. Software applications have a responsibility in the energy consumption of a device, and consequently the user, when she/he uses applications on a device, modifies the device energy behaviour. Currently there are no energy efficiency labelling mechanisms for software applications (for smartphones, tablets, desktop computers or data centres). Starting from the work of this dissertation we can create different user scenarios, which summarise high-level functionalities, for different classes of users. After that we need to define a set of labels, which will be assigned to software applications. The software application will be tested on the basis of the aforementioned scenarios, by assigning the appropriate label based on the energy consumed.

Bibliography

[1] Junit – a programmer-oriented testing framework for java, last accessed on Feb, 17, 2012.
URL http://www.junit.org/

[2] WattsUp Pro ES, last accessed on Feb, 17, 2012.
URL https://www.wattsupmeters.com/secure/products.php?pn=0

[3] Global PC, Tablet, Smartphone Sales, last accessed on January 8, 2013 (2010).
URL http://en.wiser.org/article/61670dc8dd5c5a3adba285f39bbf8145

[4] G. Abowd, A. Dey, P. Brown, N. Davies, M. Smith, P. Steggles, Towards a better understanding of context and context-awareness, in: H.-W. Gellersen (ed.), Handheld and Ubiquitous Computing, vol. 1707 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 1999, pp. 304–307.
URL http://dx.doi.org/10.1007/3-540-48157-5_29

[5] Y. Agarwal, S. Hodges, R. Chandra, J. Scott, P. Bahl, R. Gupta, Somniloquy: augmenting network interfaces to reduce pc energy usage, in: Proceedings of the 6th USENIX symposium on Networked systems design and implementation, NSDI'09, USENIX Association, Berkeley, CA, USA, 2009, pp. 365–380.
URL http://dl.acm.org/citation.cfm?id=1558977.1559002

[6] Y. Agarwal, S. Savage, R. Gupta, Sleepserver: a software-only approach for reducing the energy consumption of pcs within enterprise environments, in: Proceedings of the 2010 USENIX conference on USENIX annual technical conference, USENIXATC'10, USENIX Association, Berkeley, CA, USA, 2010, pp. 22–22.
URL http://dl.acm.org/citation.cfm?id=1855840.1855862

[7] AMD, The truth about power consumption starts here, Tech. rep., AMD (2010).
URL http://www.amd.com/us/Documents/43761C_ACP_WP_EE.pdf

[8] L. Ardito, Energy aware self-adaptation in mobile systems, 2013, pp. 1415–1417.

[9] L. Ardito, M. Morisio, Green it - available data and guidelines for reducing energy consumption in it systems, SUSTAINABLE COMPUTING.

[10] L. Ardito, G. Procaccianti, M. Torchiano, G. Migliore, Profiling power consumption on mobile devices, in: -, vol. Proceedings of The Third International Conference on Smart Grids, Green Communications and IT Energy-aware Technologies, 2013, pp. 101–106.

[11] L. Ardito, G. Procaccianti, A. Vetro', M. Morisio, Introducing energy efficiency into sqale, in: Proceedings of The Third International Conference on Smart Grids, Green Communications and IT Energy-aware Technologies, 2013, pp. 28–33.

[12] L. Ardito, M. Torchiano, M. Marengo, P. Falcarin, glcb: an energy aware context broker, Sustainable Computing: Informatics and Systems 3 (1) (2013) 18 – 26.
URL `http://dx.doi.org/10.1016/j.suscom.2012.10.005`

[13] B. Beamon, M. Kumar, Hycore: Towards a generalized hierarchical hybrid context reasoning engine, in: Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on, 2010, pp. 30 –36.

[14] D. Bein, W. Bein, S. Phoha, Efficient data centers, cloud computing in the future of distributed computing, in: Proceedings of the 2010 Seventh International Conference on Information Technology: New Generations, ITNG '10, IEEE Computer Society, Washington, DC, USA, 2010, pp. 70–75.

[15] A. Berl, E. Gelenbe, M. Di Girolamo, G. Giuliani, H. De Meer, M. Q. Dang, K. Pentikousis, Energy-efficient cloud computing, The Computer Journal 53 (7) (2009) 1045–1051.
URL `http://dx.doi.org/10.1093/comjnl/bxp080`

[16] A. P. Bianzino, C. Chaudet, D. Rossi, J.-L. Rougier, A survey of green networking research, CoRR abs/1010.3880.

[17] A. P. Bianzino, A. K. Raju, D. Rossi, Apples-to-apples: a framework analysis for energy-efficiency in networks, SIGMETRICS Perform. Eval. Rev. 38 (3) (2011) 81–85.
URL `http://doi.acm.org/10.1145/1925019.1925036`

[18] G. Brundtland, World commission on environment and development, Our common future (1987) 8–9.

[19] E. Capra, F. Merlo, Green it: Everything starts from the software., in: S. Newell, E. A. Whitley, N. Pouloudi, J. Wareham, L. Mathiassen (eds.), ECIS, 2009, pp. 62–73.
URL `http://dblp.uni-trier.de/db/conf/ecis/ecis2009.html#CapraM09`

[20] A. Carroll, G. Heiser, An analysis of power consumption in a smartphone, in: Usenix technical conference, Boston, MA, USA, 2010, pp. 1–14.

[21] L. Ceuppens, A. Sardella, D. Kharitonov, Power saving strategies and technologies in network equipment opportunities and challenges, risk and rewards, in: Proceedings of the 2008 International Symposium on Applications and the

Internet, SAINT '08, IEEE Computer Society, Washington, DC, USA, 2008, pp. 381–384.
URL http://dx.doi.org/10.1109/SAINT.2008.79

[22] W. Chedid, C. Yu, B. Lee, Power analysis and optimization techniques for energy efficient computer systems, vol. 63 of Advances in Computers, Elsevier, 2005, pp. 129 – 164.
URL http://www.sciencedirect.com/science/article/pii/S006524580463004X

[23] E. Chi, A. Salem, R. Bahar, R. Weiss, Combining software and hardware monitoring for improved power and performance tuning, in: Interaction Between Compilers and Computer Architectures, 2003. INTERACT-7 2003. Proceedings. Seventh Workshop on, 2003, pp. 57–64.

[24] L. Chiaraviglio, M. Mellia, Polisave: Efficient power management of campus pcs, in: Software, Telecommunications and Computer Networks (SoftCOM), 2010 International Conference on, 2010, pp. 82–87.

[25] Y. Chung, C. Lin, C. King, ANEPROF: Energy Profiling for Android Java Virtual Machine and Applications, Parallel and Distributed Systems, International Conference on 0 (2011) 372–379.

[26] L. Curtis, Environmentally sustainable infrastructure design, The Architecture Journal 18 (2008) 2–8.

[27] T. DeMarco, Controlling Software Projects: Management, Measurement & Estimation, Yourdon Press Computing Series, Yourdon Press, 1982.
URL http://books.google.it/books?id=y7AmAAAAMAAJ

[28] F. Ding, F. Xia, W. Zhang, X. Zhao, C. Ma, Monitoring energy consumption of smartphones, CoRR abs/1201.0218.

[29] D. Estrin, Participatory sensing: applications and architecture [internet predictions], Internet Computing, IEEE 14 (1) (2010) 12–42.

[30] P. Falcarin, M. Valla, J. Yu, C. Licciardi, C. Frà, L. Lamorte, Context data management: an architectural framework for context-aware services, Service Oriented Computing and Applications (2012) 1–18.

[31] J. Flinn, M. Satyanarayanan, Powerscope: A tool for profiling the energy usage of mobile applications, Mobile Computing Systems and Applications, IEEE Workshop on 0 (1999) 2.

[32] S. Forge, Powering down: remedies for unsustainable ICT, Foresight 9 (2007) 3–21.

[33] M. Fowler, K. Beck, Refactoring: improving the design of existing code, Addison-Wesley Professional, 1999.

[34] I. Goiri, K. Le, M. Haque, R. Beauchea, T. Nguyen, J. Guitart, J. Torres, R. Bianchini, Greenslot: Scheduling energy consumption in green datacenters, in: High Performance Computing, Networking, Storage and Analysis (SC), 2011 International Conference for, 2011, pp. 1–11.

[35] I. n. Goiri, W. Katsak, K. Le, T. D. Nguyen, R. Bianchini, Parasol and greenswitch: managing datacenters powered by renewable energy, in: Proceedings of the eighteenth international conference on Architectural support for programming languages and operating systems, ASPLOS '13, ACM, New York, NY, USA, 2013, pp. 51–64.
URL `http://doi.acm.org/10.1145/2451116.2451123`

[36] I. n. Goiri, K. Le, T. D. Nguyen, J. Guitart, J. Torres, R. Bianchini, Greenhadoop: leveraging green energy in data-processing frameworks, in: Proceedings of the 7th ACM european conference on Computer Systems, EuroSys '12, ACM, New York, NY, USA, 2012, pp. 57–70.
URL `http://doi.acm.org/10.1145/2168836.2168843`

[37] M. Gottschalk, M. Josefiok, J. Jelschen, A. Winter, Removing energy code smells with reengineering services, in: GI-Jahrestagung, 2012, pp. 441–455.

[38] T. G. Grid, Green grid metrics: Describing datacenter power efficiency (2007).

[39] S. Gude, P. Lago, a survey on green it: Metrics to express greenness in the it industry, Tech. rep., VU University (2010).

[40] S. Gude, P. Lago, Best practices for energy efficient software, http://goo.gl/AK2ho3.

[41] M. Gupta, S. Singh, Greening of the internet, in: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications, SIGCOMM '03, ACM, New York, NY, USA, 2003, pp. 19–26.
URL `http://doi.acm.org/10.1145/863955.863959`

[42] S. Gurumurthi, A. Sivasubramaniam, M. J. Irwin, N. Vijaykrishnan, M. Kandemir, T. Li, L. K. John, Using complete machine simulation for software power estimation: The softwatt approach, High-Performance Computer Architecture, International Symposium on 0 (2002) 0141.

[43] M. Halstead, R. Bayer, Algorithm dynamics, in: Proceedings of the ACM annual conference, ACM '73, ACM, New York, NY, USA, 1973, pp. 126–135.
URL `http://doi.acm.org/10.1145/800192.805693`

[44] J. Hamilton, Cooperative expendable micro-slice servers (cems): low cost, low power servers for internet-scale services, 4th Biennial Conference on Innovative Data Systems Research (CIDR) Asilomar.

[45] S. Herat, Sustainable management of electronic waste (e-waste), CLEAN Soil, Air, Water 35 (4) (2007) 305–310.
URL `http://dx.doi.org/10.1002/clen.200700022`

[46] M. Hollander, D. A. Wolfe, Nonparametric Statistical Methods, John Wiley & Sons, New York, 1973.

[47] A. Hylick, R. Sohan, A. Rice, B. Jones, An analysis of hard drive energy consumption, in: MASCOTS, 2008, pp. 103–112.

[48] IDC, International data corporation, document no. 221346 (2009).

[49] ISO/IEC, 9126. Software engineering – Product quality, ISO/IEC, 2001.

[50] ISO/IEC, 25010. Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models, ISO/IEC, 2011.

[51] A. Jaiantilal, Y. Jiang, S. Mishra, Modeling cpu energy consumption for energy efficient scheduling, in: Proceedings of the 1st Workshop on Green Computing, GCM '10, ACM, New York, NY, USA, 2010, pp. 10–15.
URL `http://doi.acm.org/10.1145/1925013.1925015`

[52] N. Juristo, A. M. Moreno, Basics of Software Engineering Experimentation, 1st ed., Springer Publishing Company, Incorporated, 2010.

[53] G. Kaefer, Green sw engineering: Ideas for including energy efficiency into your software projects, Post Proceedings ICSE 2009.

[54] A. Kansal, F. Zhao, J. Liu, N. Kothari, A. A. Bhattacharya, Virtual machine power metering and provisioning, in: Proceedings of the 1st ACM symposium on Cloud computing, SoCC '10, ACM, New York, NY, USA, 2010, pp. 39–50.
URL `http://doi.acm.org/10.1145/1807128.1807136`

[55] F. Khomh, M. Penta, Y. Guéhéneuc, G. Antoniol, An exploratory study of the impact of antipatterns on class change-and fault-proneness, Empirical Software Engineering (2012) 1–33.

[56] N. Kiyanclar, A survey of virtualization techniques focusing on secure on-demand cluster computing (2005).

[57] J. Koomey, Growth in data center electricity use 2005 to 2010, Tech. rep., Stanford University (2011).
URL `http://www.analyticspress.com/datacenters.html`

[58] R. Kravets, P. Krishnan, Power management techniques for mobile communication, in: Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking, MobiCom '98, ACM, New York, NY, USA, 1998, pp. 157–168.
URL `http://doi.acm.org/10.1145/288235.288276`

[59] J. Krikke, Recycling e-waste: The sky is the limit, IT Professional 10 (1) (2008) 50–55.

[60] L. Lamorte, C. A. Licciardi, M. Marengo, A. Salmeri, P. Mohr, G. Raffa, L. Roffia, M. Pettinari, S. T. Cinotti, A platform for enabling context aware telecommunication services, Third Workshop on Context Awareness for Proactive Systems.

[61] P. Larsson, Energy-efficient software guidelines (2011) 14.
URL `http://software.intel.com/file/1332`

[62] J.-L. Letouzey, T. Coq, The sqale analysis model: An analysis model compliant with the representation condition for assessing the quality of software source code, in: Advances in System Testing and Validation Lifecycle (VALID), 2010 Second International Conference on, 2010, pp. 43 –48.

[63] T. Li, L. K. John, Run-time modeling and estimation of operating system power consumption, in: Proceedings of the 2003 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, SIGMETRICS '03, ACM, New York, NY, USA, 2003, pp. 160–171.
URL http://doi.acm.org/10.1145/781027.781048

[64] J. Lorch, A complete picture of the energy consumption of a portable computer, Tech. rep. (1995).

[65] J. Lorch, A. Smith, Software strategies for portable computer energy management, Personal Communications, IEEE 5 (3) (1998) 60 –73.

[66] E. Macii, M. Pedram, F. Somenzi, High-level power modeling, estimation, and optimization, Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on 17 (11) (1998) 1061 –1079.

[67] P. Mahadevan, P. Sharma, S. Banerjee, P. Ranganathan, A power benchmarking framework for network devices, in: Proceedings of the 8th International IFIP-TC 6 Networking Conference, NETWORKING '09, Springer-Verlag, Berlin, Heidelberg, 2009, pp. 795–808.
URL http://dx.doi.org/10.1007/978-3-642-01399-7_62

[68] T. McCabe, A complexity measure, IEEE Trans. Soft. Eng. SE-2 (4) (1976) 308 – 320.

[69] G. McCluskey, Tuning Java I/O Performance, last accessed on December 29, 2013 (1999).
URL http://goo.gl/7NKUhv

[70] D. Meisner, B. T. Gold, T. F. Wenisch, Powernap: eliminating server idle power, SIGPLAN Not. 44 (3) (2009) 205–216.
URL http://doi.acm.org/10.1145/1508284.1508269

[71] A. Molla, Organizational motivations for green it: Exploring green it matrix and motivation models, in: PACIS, 2009, p. 13.

[72] V. G. Moshnyaga, Integrated circuit and system design. power and timing modeling, optimization and simulation, 2009, pp. 82–92.
URL http://dx.doi.org/10.1007/978-3-540-95948-9_9

[73] V. G. Moshnyaga, A new approach for energy management in user-centric applications, in: Proceedings of the International Conference on Green Computing, GREENCOMP '10, IEEE Computer Society, Washington, DC, USA, 2010, pp. 107–112.
URL http://dx.doi.org/10.1109/GREENCOMP.2010.5598257

[74] S. Murugesan, Harnessing green it: Principles and practices, IT Professional 10 (1) (2008) 24–33.
URL http://dx.doi.org/10.1109/MITP.2008.10

[75] S. Murugesan, Making it green, IT Professional 12 (2) (2010) 4–5.

[76] S. Naumann, M. Dick, E. Kern, T. Johann, The greensoft model: A reference model for green and sustainable software and its engineering, Sustainable

Computing Informatics and Systems 1 (4) (2011) 294–304.
URL http://linkinghub.elsevier.com/retrieve/pii/S2210537911000473

[77] B. Nordman, K. Christensen, Reducing the energy consumption of network devices, IEEE 8023 tutorial (2005) 1–30.

[78] R. Palit, R. Arya, K. Naik, A. Singh, Selection and execution of user level test cases for energy cost evaluation of smartphones, in: Proceedings of the 6th International Workshop on Automation of Software Test, 2011, pp. 84–90.

[79] A. Pathak, Y. C. Hu, M. Zhang, Bootstrapping energy debugging on smartphones: a first look at energy bugs in mobile devices, in: Proceedings of the 10th ACM Workshop on Hot Topics in Networks, HotNets-X, ACM, New York, NY, USA, 2011, pp. 5:1–5:6.
URL http://doi.acm.org/10.1145/2070562.2070567

[80] J. Peddersen, S. Parameswaran, Energy driven application selfadaptation, in: VLSI Design, 2007. Held jointly with 6th International Conference on Embedded Systems., 20th International Conference on, 2007, pp. 385–390.

[81] G. Procaccianti, L. Ardito, A. Vetrò, M. Morisio, Energy efficiency in the ict - profiling power consumption in desktop computer systems, in: Energy Efficiency - The Innovative Ways for Smart Energy, the Future Towards Modern Utilities / InTech, Prof. Moustafa Eissa, Helwan, 2012, pp. 353–372.

[82] G. Procaccianti, A. Vetrò, L. Ardito, M. Morisio, Profiling power consumption on desktop computer systems, LECTURE NOTES IN COMPUTER SCIENCE 6868 (2011) 110–123.

[83] S. Rivoire, M. A. Shah, P. Ranganathan, C. Kozyrakis, Joulesort: a balanced energy-efficiency benchmark, in: Proceedings of the 2007 ACM SIGMOD international conference on Management of data, SIGMOD '07, ACM, New York, NY, USA, 2007, pp. 365–376.
URL http://doi.acm.org/10.1145/1247480.1247522

[84] S. Rivoire, M. A. Shah, P. Ranganathan, C. Kozyrakis, J. Meza, Models and metrics to enable energy-efficiency optimizations, Computer 40 (2007) 39–48.

[85] J. A. Roberson, C. A. Webber, M. C. McWhinney, R. E. Brown, M. J. Pinckard, J. F. Busch, After-hours power status of office equipment and inventory of miscellaneous plug-load equipment (2004) 1–33.

[86] K. W. Roth, F. Goldstein, J. Kleinman, Energy Consumption by Office and Telecommunications Equipment in Commercial Buildings Volume I: Energy Consumption Baseline, Tech. rep., US Department of Energy (2002).

[87] J. T. Russell, M. F. Jacome, Software power estimation and optimization for high performance, 32-bit embedded processors, in: Proceedings of the International Conference on Computer Design, ICCD '98, IEEE Computer Society, Washington, DC, USA, 1998, pp. 328–.
URL http://dl.acm.org/citation.cfm?id=846214.846614

[88] E. Saxe, Power-efficient software, Queue 8 (1) (2010) 10:10–10:17.
URL `http://doi.acm.org/10.1145/1698223.1698225`

[89] J. Scaramella, Worldwide Server Energy Expense 2009–2013 Forecast, Tech. rep., International Data Corporation (IDC), Document No. 221346 (2009).
URL `www.idc.com`

[90] N. Sharma, S. Barker, D. Irwin, P. Shenoy, Blink: managing server clusters on intermittent power, in: Proceedings of the sixteenth international conference on Architectural support for programming languages and operating systems, ASPLOS XVI, ACM, New York, NY, USA, 2011, pp. 185–198.
URL `http://doi.acm.org/10.1145/1950365.1950389`

[91] F. Shull, J. Singer, D. I. Sjøberg, Guide to Advanced Empirical Software Engineering, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

[92] R. Singh, D. Irwin, P. Shenoy, K. K. Ramakrishnan, Yank: enabling green data centers to pull the plug, in: Proceedings of the 10th USENIX conference on Networked Systems Design and Implementation, nsdi'13, USENIX Association, Berkeley, CA, USA, 2013, pp. 143–156.
URL `http://dl.acm.org/citation.cfm?id=2482626.2482642`

[93] A. Sinha, A. Chandrakasan, Jouletrack-a web based tool for software energy profiling, in: Design Automation Conference, 2001. Proceedings, 2001, pp. 220 – 225.

[94] P. Somavat, S. Jadhav, V. Namboodiri, Accounting for the energy consumption of personal computing including portable devices, in: Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking, e-Energy '10, ACM, New York, NY, USA, 2010, pp. 141–149.
URL `http://doi.acm.org/10.1145/1791314.1791337`

[95] B. Steigerwald, R. Chabukswar, K. Krishnan, J. De Vega, Creating energy - efficient software, Context 184 (1) (2007) 30.

[96] P.-N. Tan, M. Steinbach, V. Kumar, Introduction to Data Mining, (First Edition), Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.

[97] V. Tiwari, M. T.-C. Lee, Power analysis of a 32-bit embedded microcontroller, in: Proceedings of the 1995 Asia and South Pacific Design Automation Conference, ASP-DAC '95, ACM, New York, NY, USA, 1995.
URL `http://doi.acm.org/10.1145/224818.224890`

[98] V. Tiwari, S. Malik, A. Wolfe, Power analysis of embedded software: a first step towards software power minimization, Very Large Scale Integration (VLSI) Systems, IEEE Transactions on 2 (4) (1994) 437 –445.

[99] R. Van Solingen, E. Berghout, Goal/Question/Metric Method, McGraw-Hill Inc.,US, 1999.

[100] A. Vetro', L. Ardito, M. Morisio, G. Procaccianti, Monitoring it power consumption in a research center: Seven facts, in: Proceedings of ENERGY 2011,

The First International Conference on Smart Grids, Green Communications and IT Energy-aware Technologies, 2011, pp. 64–69.

[101] A. Vetro', L. Ardito, G. Procaccianti, M. Morisio, Definition, implementation and validation of energy code smells: an exploratory study on an embedded system, in: Proceedings of ENERGY 2013 : The Third International Conference on Smart Grids, Green Communications and IT Energy-aware Technologies, 2013, pp. 34–39.

[102] Y. Wati, C. Koo, The green it practices of nokia, samsung, sony, and sony ericsson: Content analysis approach, 2013 46th Hawaii International Conference on System Sciences 0 (2010) 1–10.

[103] M. Webb, E. Al., Smart 2020: Enabling the low carbon economy in the information age, The Climate Group London.
URL `http://www.ecoc2008.org/documents/SYMPTu_Webb.pdf`

[104] C. A. Webber, J. A. Roberson, M. C. McWhinney, R. E. Brown, M. J. Pinckard, J. F. Busch, After-hours power status of office equipment in the usa, Energy 31 (14) (2006) 2823 – 2838.

[105] E. Williams, Energy intensity of computer manufacturing: hybrid assessment combining process and economic input-output methods., Environmental science technology 38 (22) (2004) 6166–6174.
URL `http://pubs.acs.org/doi/abs/10.1021/es035152j`

[106] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, A. Wesslén, Experimentation in software engineering: an introduction, Kluwer Academic Publishers, Norwell, MA, USA, 2000.

[107] N. Zazworka, M. Shaw, F. Shull, C. Seaman, Investigating the impact of design debt on software quality, in: Proceeding of the 2nd workshop on Managing technical debt, ACM, 2011, pp. 17–23.

[108] L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R. P. Dick, Z. M. Mao, L. Yang, Accurate online power estimation and automatic battery behavior based power model generation for smartphones, in: Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis, CODES/ISSS '10, ACM, New York, NY, USA, 2010, pp. 105–114.